Tampereen yliopisto
Tampere University

COMP.SE.100-EN ItSE

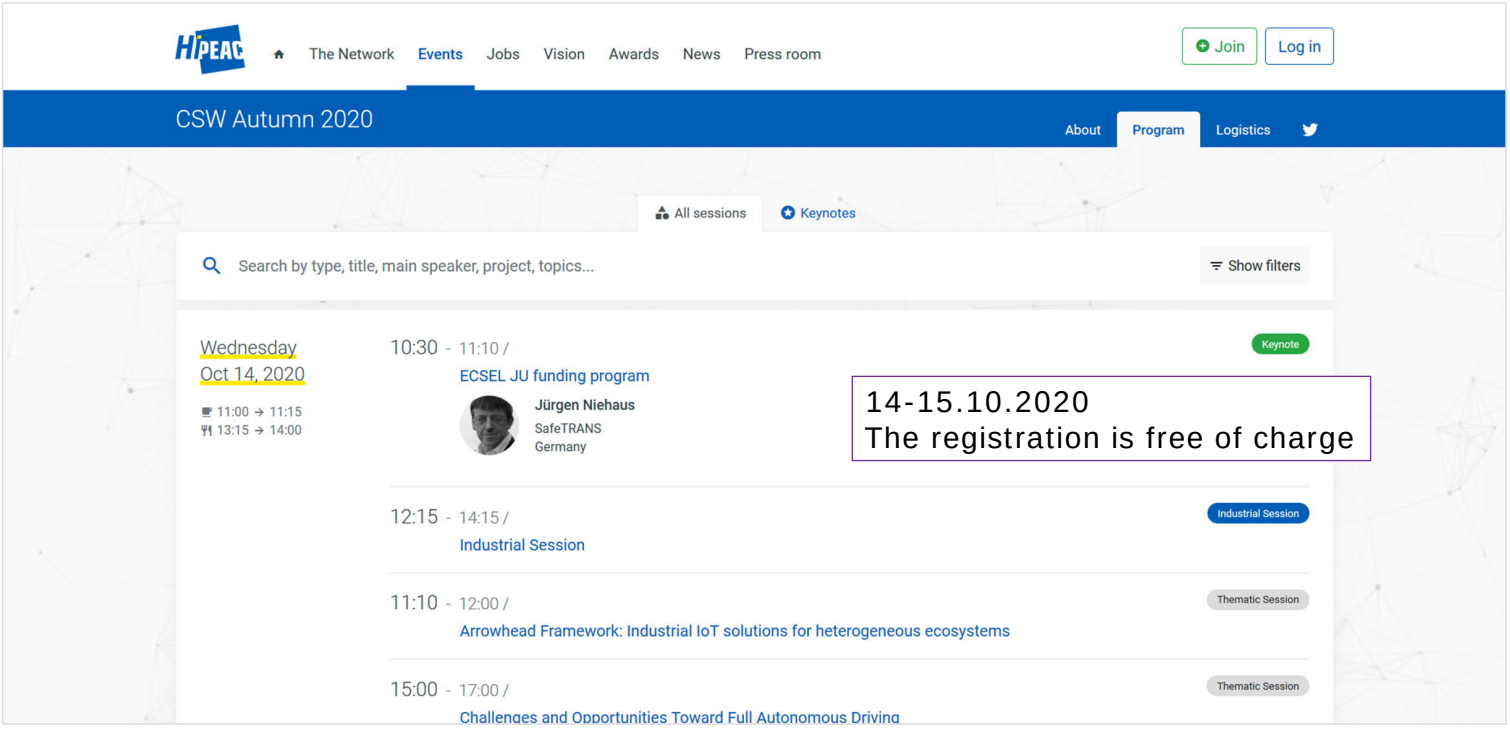
Zoom begins soon...

at 1415 o'clock.

Tampereen yliopisto
Tampere University

HiPEAC, European Network on High-performance Embedded Architecture and Compilation

[https://www.hipeac.net/csw/2020/autumn-virtual/#/program/]



14-15.10.2020
The registration is free of charge

Online courses available; https://www.y-kampus.fi/en/studies/

COMP.SE.100-EN, 2020, course schedule v6c (02.09.2020)

week	lectures	exam	weekly exercises	project assignment (exercise work)	week
35	L1: course basics		--- sign to WE groups ---	sign for project = grouping...	35
36	Project Assignment explained		**WE1**: intro to requirements	grouping, groups to Moodle	36
37	L2: Sw Eng in general		**WE2**: Trello and agile way	group's Trello board ready with product backlog	37
38	L3: requirements		**WE3**: feasibility study and stakeholder analysis	working...	38
39	L4: basic UML diagrams		**WE4**: requirements	working...	39
40	L5: more UML diagrams	EXAM-1	**WE5**: UML diagrams - Use case	working...	40
41	L6: different sw systems	EXAM-1	**WE6**: UML diagrams - concept/entity and navigation	deadline for 1st phase documentation and presentation	41
42	examination week		examination week	examination week	42
43	L7: life cycle models		groups' 1st presentations	groups' 1st phase presentations	43
44	L8: quality and testing	EXAM-2	**WE7**: development processes	feedback group-to-group at PRP, from 1st phase	44
45	L9: project work	EXAM-2	**WE8**: testing and error reporting	deadline for diagrams first versions (Moodle)	45
46	L10: project management		**WE9**: effort estimation	feedback to groups from diagrams (from assistants)	46
47	L11: open source, APIs, IPR		**WE10**: delivery contracts and terms of use	deadline for 2nd phase presentation (PRP)	47
48	L12: embedded systems, IoT	EXAM-3	groups' final presentations	groups' final presentations / feedback g-to-g (PRP)	48
49	L13: recap, summary	EXAM-3	- - -	final (2.) delivery of project documentation	49
50	examination week		examination week	feedback inside group, student-to-student at PRP	50
51	examination week		examination week	end of game / game over.	51
	Lectures: Wed at 1415-16.		Weekly exercises:		
			Mon 0815-10	AUTUMN 2020 (1-2. periods)	
			Mon 1215-14	are remote/distant learning.	
			Tue 0815-10		
			Tue 1415-16		
			Wed 0815-10.		

Remote/distant learning 2020.
No contact teaching at ItSE 2020.

Tampereen yliopisto
Tampere University

COMP.SE.100 -EN "ItSE"

Introduction to Software Engineering

2020, 1-2. periods

5 credit units

06-swsys-ItSE-2020-v6

Tampereen yliopisto
Tampere University

COMP.SE.100-EN (ItSE)
Introduction to Software Engneering

Lecture 6, 07.10.2020

Tensu: remember to start Zoom lecture recording, at 1415

Prefer course Moodle over SISU information.

Students are recommended to follow Moodle News/messages.

Tampereen yliopisto
Tampere University

Face mask recommendation update in Tampere region: always wear a face mask when on campus

The face mask recommendation for the Tampere region has been updated on 6 October and now extends to higher education establishments. Tampere University and TAMK instruct wearing a face mask at all times when on campus.

According to a press release by Pirkanmaa Hospital District published on 6 October (in Finnish), the face mask recommendation in Tampere region has been updated because the COVID-19 incidence in the Tampere region has recently been highest among persons aged 20–29 years. In addition, more than ten cases have been identified among higher education students in Tampere region, with links to student events in Vaasa.

Starting from 6 October, Tampere University and TAMK instruct **wearing a face mask at all times when on campus**. Coming to the campus when feeling unwell in not allowed. In addition to wearing a face mask, all campus users must observe the safety precautions given previously (a safe distance from others, restrictions on gatherings, proper hygiene practices).

To protect the community, it is important to also observe the safety precautions during free time. If you have respiratory symptoms, get tested for Covid-19.

Tampere University and TAMK strongly recommend all members of the community and those visiting the campuses **download and use the Koronavilkku contact tracing app**.

Tampereen yliopisto
Tampere University

COMP.SE.100-EN, 2020, course schedule v6c (02.09.2020)

week	lectures	exam	weekly exercises	project assignment (exercise work)	week
35	L1: course basics		--- sign to WE groups ---	sign for project = grouping...	35
36	Project Assignment explained		**WE1**: intro to requirements	grouping, groups to Moodle	36
37	L2: Sw Eng in general		**WE2**: Trello and agile way	group's Trello board ready with product backlog	37
38	L3: requirements		**WE3**: feasibility study and stakeholder analysis	working...	38
39	L4: basic UML diagrams		**WE4**: requirements	working...	39
40	L5: more UML diagrams	EXAM-1	**WE5**: UML diagrams - Use case	working...	40
41	L6: different sw systems	EXAM-1	**WE6**: UML diagrams - concept/entity and navigation	deadline for 1st phase documentation and presentation	41
42	examination week		examination week	examination week	42
43	L7: life cycle models		groups' 1st presentations	groups' 1st phase presentations	43
44	L8: quality and testing	EXAM-2	**WE7**: development processes	feedback group-to-group at PRP, from 1st phase	44
45	L9: project work	EXAM-2	**WE8**: testing and error reporting	deadline for diagrams first versions (Moodle)	45
46	L10: project management		**WE9**: effort estimation	feedback to groups from diagrams (from assistants)	46
47	L11: open source, APIs, IPR		**WE10**: delivery contracts and terms of use	deadline for 2nd phase presentation (PRP)	47
48	L12: embedded systems, IoT	EXAM-3	groups' final presentations	groups' final presentations / feedback g-to-g (PRP)	48
49	L13: recap, summary	EXAM-3	- - -	final (2.) delivery of project documentation	49
50	examination week		examination week	feedback inside group, student-to-student at PRP	50
51	examination week		examination week	`end of game / game over.`	51
	Lectures: Wed at 1415-16.		Weekly exercises:		
			Mon 0815-10	AUTUMN 2020 (1-2. periods)	
			Mon 1215-14	are remote/distant learning.	
			Tue 0815-10		
			Tue 1415-16		
			Wed 0815-10.		

Tampereen yliopisto
Tampere University

A million-row limit on Microsoft's Excel spreadsheet software may have led to Public Health England misplacing nearly 16,000 Covid test results, it is understood. The data error, which led to 15,841 positive tests being left off the official daily figures, means than 50,000 potentially infectious people may have been missed by contact tracers and not told to self-isolate.

But while CSV files can be any size, Microsoft Excel files can only be 1,048,576 rows long – or, in older versions which PHE may have still been using, a mere 65,536. When a CSV file longer than that is opened, the bottom rows get cut off and are no longer displayed.

Tampereen yliopisto
Tampere University

First, general course matters

Juanita: groups G01-G04

Aleksius: ODD groups; G05,G07,G09,G11,G13,G15,G17,G19,G21,G23,G27

Lauri: EVEN groups; G06,G08,G10,G12,G14,G16,G18,G20,G22,G24,G28

- Trello board is used as help for work division and assignment

WE attendees:

- **Mon 0815-10** 9, 8,10, 5, 6, 4,
- **Mon 1215-14** 11,12,12,13,11,12,
- **Tue 0815-10** 3, 6, 4, 6, 5, 5,
- **Tue 1415-16** 8,10, 9, 8, 5, 7,
- **Wed 0815-10** 12,11, 9, 8, 7, 6,

Very small WEs are not reasonable, we think how many groups will continue at 2nd period…

Voting at Moodle, ends 14.10.

Tampereen yliopisto
Tampere University

Current at course (w 41)

- WE6 was about other useful diagrams needed at PA and EXAM

- EXAM 1/3 (w40-41) is ongoing, 16 taken, 36 reservations so far, do not leave your EXAM reservation for the last days

- for WE5 and WE6, install Dia to your own computer
- (EXAM classes have Dia and EA tools installed)

- remember to use (i.e. update) your group's Trello board.

Tampereen yliopisto
Tampere University

Current at course (w 41)

- 1st presentation times; Aleksius' and Lauri's groups will be available for reservations at Moodle from 07.10.2020 at 1600 o'clock (today). 3..4 project groups at one presentation session.

- weekly exercise groups for 2nd study period; at Moodle there is a new voting, current five groups seem to be too much for some 30 students.

Tampereen yliopisto
Tampere University

Potential problem; ”sleepers” or ”free-riders” at project group ?

Some hints

Tampereen yliopisto
Tampere University

Course contents (plan)

1. Course basics, intro
2. Sw Eng in general, overview
3. Requirements
4. **Basic UML Diagrams (”Class”, Use Case, Navigation)**
5. UML diagrams, in more detail
6. Different software systems
7. Life Cycle models
8. Quality and Testing
9. Project work
10. Project management
11. Open source, APIs, IPR
12. Embedded systems
13. Recap

Tampereen yliopisto
Tampere University

6. Different kind of software systems

- Different kind of software systems; by architecture, purpose or development
  - stand-alone
  - client-server
  - mobile
  - web and PWA
  - cross-platform
  - real-time
  - reactive.
- SaaS, IaaS, Paas, etc. XaaS
- business, technical/engineering/scientific,...
- mission-critical / safety-critical systems (more at L12)
- distributed (GSD = global software development) (more at L8)
- architecture.

(Embedded and real-time systems on L12, IoT on L11.)

Tampereen yliopisto
Tampere University

Several classifications of sw systems

Basis of application [geeksforgeeks.org]:

- system software / network or web application / embedded / business / entertainment or gaming / artifical intelligence / scientific / document management / utilities.

Basis of copyright [geeksforgeeks.org]:

- commerical / shareware / freeware / public domain / open source.

Classification of software [ecomputernotes.com]

- system sw / real-time sw / business sw / engineering and scientific / artifical intelligence / web-based / personal computer.

Basis of deployment platform...

- PC, mobile, clitent-server, cloud, hosted (XaaS),...

Software system types, 1

- System software
  - e.g. operating systems, tools
- Mobile application
  - e.g. GSM, tablet
- Organisational information systems
  - application programs
- Network services
  - e.g. internet, www.

FI: sovelma = applet (www-selaimessa toimiva ohjelma)
FI: sovellus = application.

Software system types, 2

- Embedded systems
  - inside a machine or device
    - e.g. lift/elevator control system
- Real-time systems
  - software must react immediately
    - e.g. fuel and breaks control in cars
- Reactive systems
  - operate continuously
    - e.g. GSM switch/exchange centre.

Almost all devices nowadays contain software; refrigerator, door lock, coffee machine, lamp,...

Terminology, ISO/IEC/IEEE 24765:2017 standard

3.3829 , software product

1. set of computer programs, procedures, and possibly associated documentation and data
2. any of the individual items in (1)
3. complete set of software designed for delivery to a software consumer or end-user, which can include computer programs, procedures and associated documentation and data.
4. set of computer programs, procedures, database- and other data structure descriptions and associated documentation

cf. software package

Note 1 to entry: A software product can be designated for delivery, an integral part of another product, or used in development. Software products vary from large customized application software for one customer to standard software packages that are sold off the shelf to millions of customers.

Tampereen yliopisto
Tampere University

Terminology, ISO/IEC/IEEE 24765:2017 standard

3.216 , architecture

1. [system] fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution
2. set of rules to define the structure of a system and the interrelationships between its parts

cf. component, module, subprogram, routine.

3.3870 , software/system element

1. element that defines and prescribes what a software or system is composed of

cf. software element

EXAMPLE: requirements, design, code, test cases, and version number.

Terminology, ISO/IEC/IEEE 24765:2017 standard

3.4090 , system

1. combination of interacting elements organized to achieve one or more stated purposes
2. product of an acquisition process that is delivered to the user
3. something of interest as a whole or as comprised of parts
4. interacting combination of elements to accomplish a defined objective
5. set of interrelated or interacting elements

Note 1 to entry: A system is sometimes considered as a product or as the services it provides. In practice, the interpretation of its meaning is frequently clarified by the use of an associative noun, e.g., aircraft system. Alternatively, the word 'system' can be replaced by a context-dependent synonym, e.g., aircraft, though this obscures the system perspective. A complete system includes all of the associated equipment, facilities, material, computer programs, firmware, technical documentation, services, and personnel required for operations and support to the degree necessary for self-sufficient use in its intended environment.

Once again, alternative terminology may exist...

stand-alone = single PC and data processing happens there.

client-server = server does data processing, and returns results to client (GUI). Database (server) may be ”3rd layer”.

mobile = any mobile device; www application (program) or applet (requires browser).

web and PWA (progressive web application)

cross-platform (mobile) = code is written once and modified by a tool for many platforms

JAVA lang is suitable to this

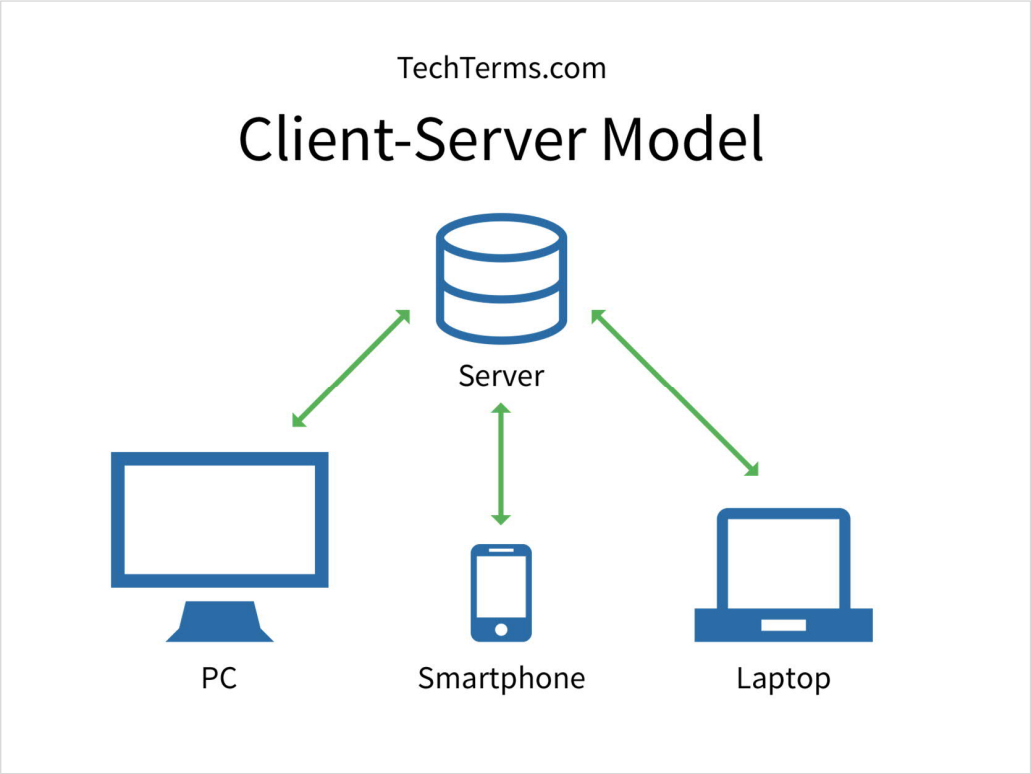
Most common tools (and many more exists...)

- ReactNative (Facebook)
- Xamarin (Microsoft)
- Flutter (Google).

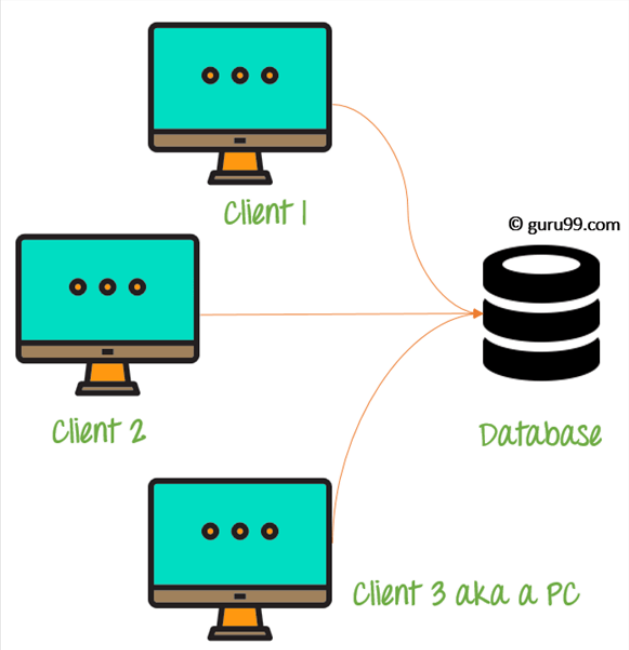
multi-platform = the same code runs on many platforms.

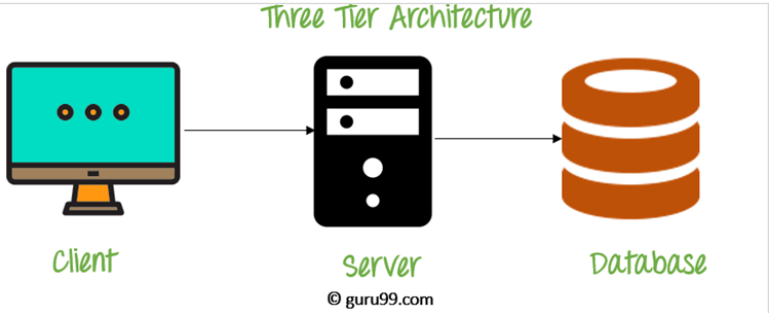
Client part has only GUI and minimal amount of processing. Heavy data procssing is done at the server.

By the way, ”full-stack developer” can code both client and server software.



Client-server





Tampereen yliopisto
Tampere University

[https://www.geeksforgeeks.org/client-server-model/]

Advantages of Client-Server model:

Centralized system with all data in a single place.

Cost efficient requires less maintenance cost and Data recovery is possible.

The capacity of the Client and Servers can be changed separately.

Disadvantages of Client-Server model:

Clients are prone to viruses, Trojans and worms if present in the Server or uploaded into the Server.

Server are prone to Denial of Service (DOS) attacks.

Data packets may be spoofed or modified during transmission.

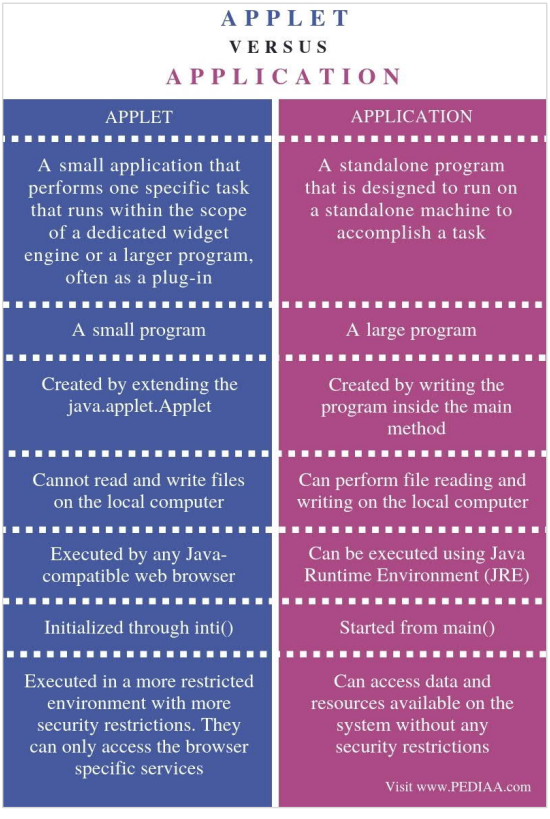
Phishing or capturing login credentials or other useful information of the user are common and MITM (Man in the Middle) attacks are common.

Tampereen yliopisto
Tampere University

Application vs. applet

Java Application	Applet
Java application contains a main method	An applet does not contain a main method
Does not require internet connection to execute	Requires internet connection to execute
Is stand alone application	Is a part of web page
Can be run without a browser	Requires a Java compatible browser
Uses stream I/O classes	Use GUI interface provided by AWT or Swings
Entry point is main method	Entry point is init method
Generally used for console programs	Generally used for GUI interfaces

[https://www.startertutorials.com/]



Edge computing

Edge computing (FI: reunalaskenta)

Gartner defines edge computing as “a part of a distributed computing topology in which information processing is located close to the edge – where things and people produce or consume that information.”

At its basic level, edge computing brings computation and data storage closer to the devices where it’s being gathered, rather than relying on a central location that can be thousands of miles away. This is done so that data, especially real-time data, does not suffer latency issues that can affect an application’s performance. In addition, companies can save money by having the processing done locally, reducing the amount of data that needs to be processed in a centralized or cloud-based location.

Edge computing was developed due to the exponential growth of IoT devices, which connect to the internet for either receiving information from the cloud or delivering data back to the cloud. And many IoT devices generate enormous amounts of data during the course of their operations.

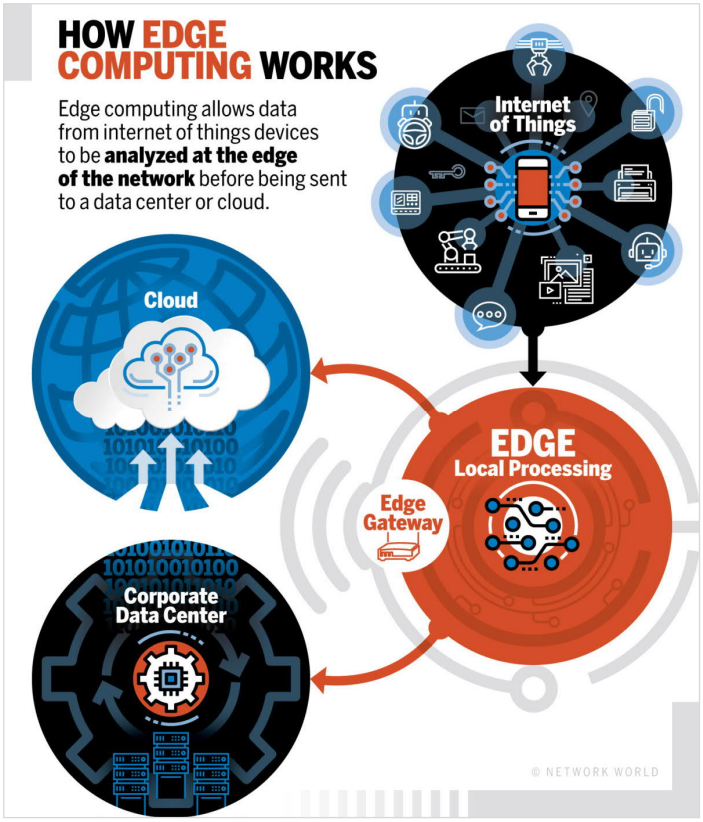
[https://www.networkworld.com/]

Tampereen yliopisto
Tampere University

Edge computing

Around the world, carriers are deploying 5G wireless technologies, which promise the benefits of high bandwidth and low latency for applications, enabling companies to go from a garden hose to a firehose with their data bandwidth. Instead of just offering the faster speeds and telling companies to continue processing data in the cloud, many carriers are working edge-computing strategies into their 5G deployments in order to offer faster real-time processing, especially for mobile devices, connected cars and self-driving cars.

[https://www.networkworld.com/]

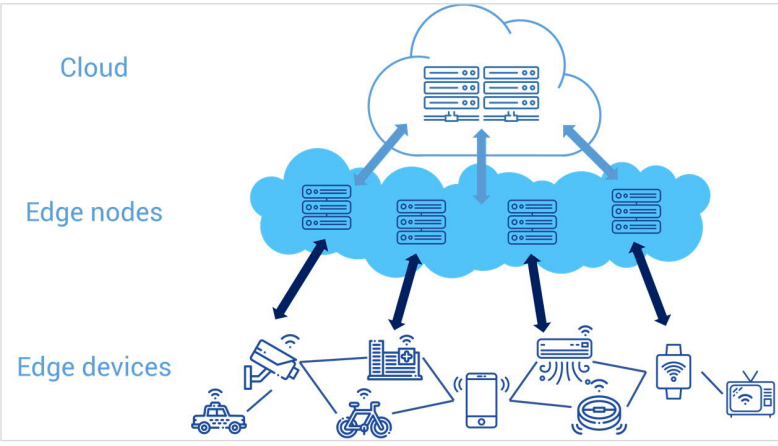


Tampereen yliopisto
Tampere University

Edge computing is a distributed computing concept that integrates intelligence to edge devices, also called edge nodes, allowing data to be processed and analyzed in real time near the data collection source.

In edge computing, data does not need to be uploaded directly to the cloud or to a centralized data processing system.

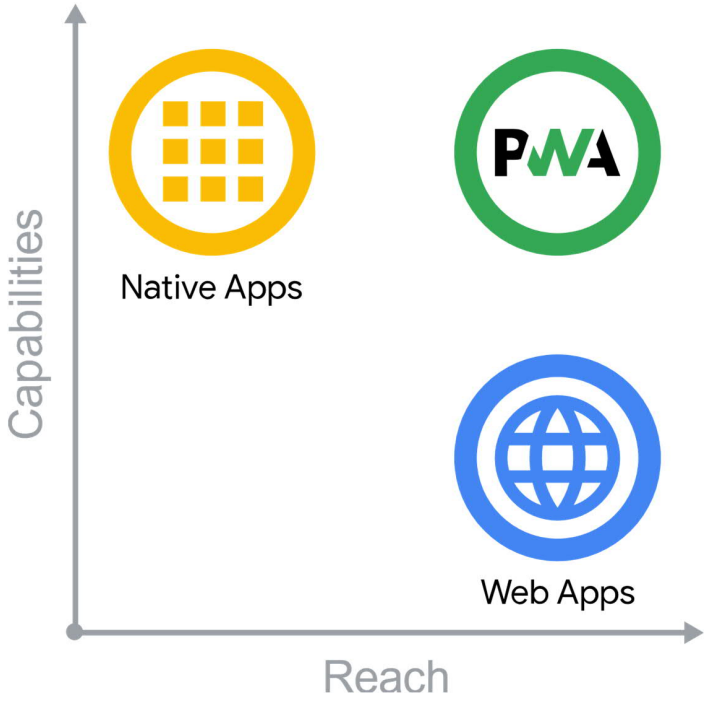
The main difference between cloud computing and edge computing is where the data is being processed. In cloud computing, data is collected, processed, and analyzed at a centralized location. On the other hand, edge computing is based on a distributed computing environment, in which data is collected, processed, and analyzed locally.



[https://www.alibabacloud.com/knowledge/what-is-edge-computing]

PWA
Progressive web application

PWA [https://web.dev/what-are-pwas/]



Tampereen yliopisto
Tampere University

PWA [https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps]

Progressive Web Apps are web apps that use emerging web browser APIs and features along with traditional progressive enhancement strategy to bring a native app-like user experience to cross-platform web applications.

Progressive Web Apps are a useful design pattern, though they aren't a formalized standard. PWA can be thought of as similar to AJAX or other similar patterns that encompass a set of application attributes, including use of specific web technologies and techniques. This set of docs tells you all you need to know about them.

In order to call a Web App a PWA, technically speaking it should have the following features: Secure contexts (HTTPS), one or more Service Workers, and a manifest file.

PWAs are web apps developed using a number of specific technologies and standard patterns to allow them to take advantage of both web and native app features.

PWAs should be discoverable, installable, linkable, network independent, progressive, re-engageable, responsive, and safe.

Tampereen yliopisto
Tampere University

Cross-platform development (mobile)

Tampereen yliopisto
Tampere University

Cross-platform vs. multi-platform development

Cross-platform or Multi-platform software may be divided into two types;

- One requires individual building or compilation for each platform that it supports. This means that the app is provided with different builds for different platforms.
- And the other one can be directly run on any platform without special preparation. So, the app is provided by a single package that can be built for all platforms.

Generally, you do cross-platform development when you use a tool that allows you to write code once and built it for many platforms.

While multi-platform means your code runs on many platforms.

[https://luismts.com/maui-cross-platform-vs-multi-platform/]

Tampereen yliopisto
Tampere University

What Is Cross Platform Development, 1

Normally when you build a native app you have to build a separate one for Android and a separate one for iOS/Apple, each using that specific platform language.

For example, to build for the following platforms you need to use the standard language they recognize:

- Apple iOS: Program in Objective C or Swift
- Android: Program in Java
- Windows Phone: Program in C# and XAML.

[https://www.cleart.com/what-is-cross-platform-development.html]

Native code means you program just for the one target operating system, and take advantage of the mobile device's hardware characteristics. Performance is fast and versatile (e.g. graphics). But the code does not run on other kind of devices.

What Is Cross Platform Development, 2

Unfortunately the need for separate platform languages has the following disadvantages:

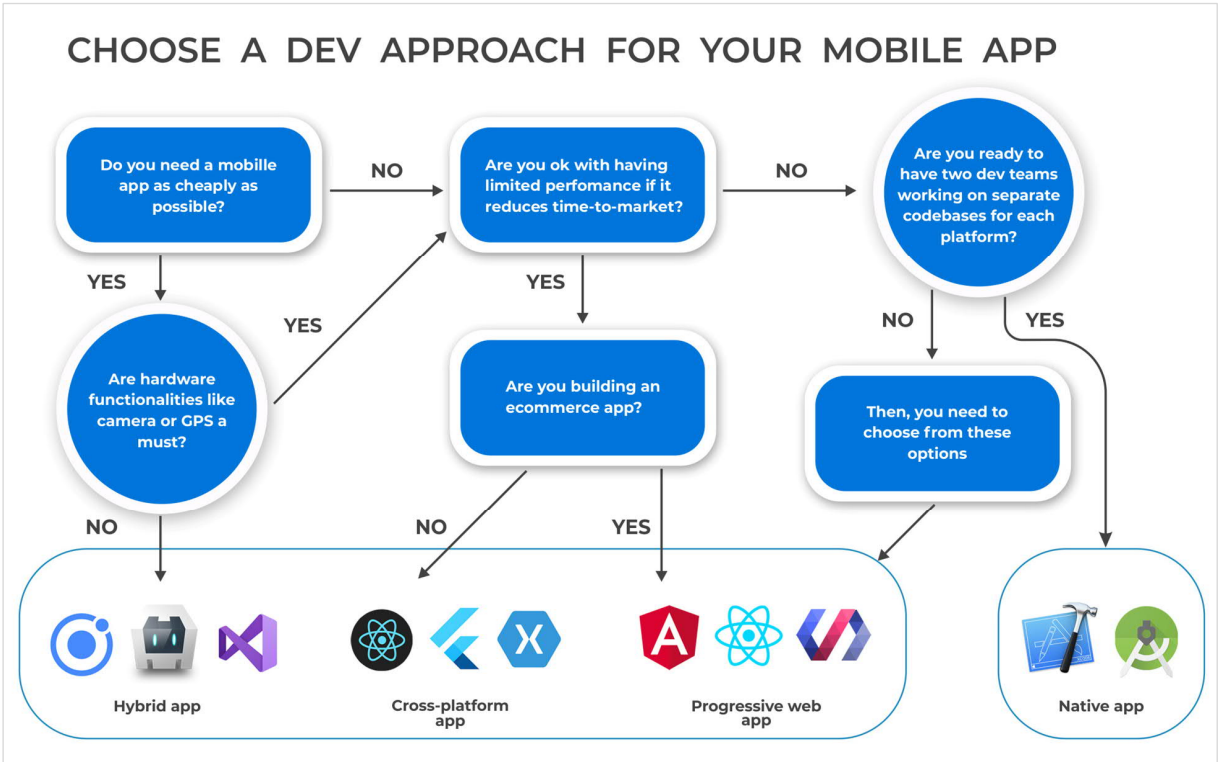
Expense: Creating and maintaining an app for each operating system is much more expensive.

Knowledge: Programming native mobile apps would require high levels of skill in 2-3 different languages.

Uniformity: Because each platform has its own user interface, standardized widgets and features, apps will not be uniform from platform to platform and will create a different user experience depending on the device. Many people have more than one type of device so when they use your app on iOS and switch over to an Android device, their experience will be different.

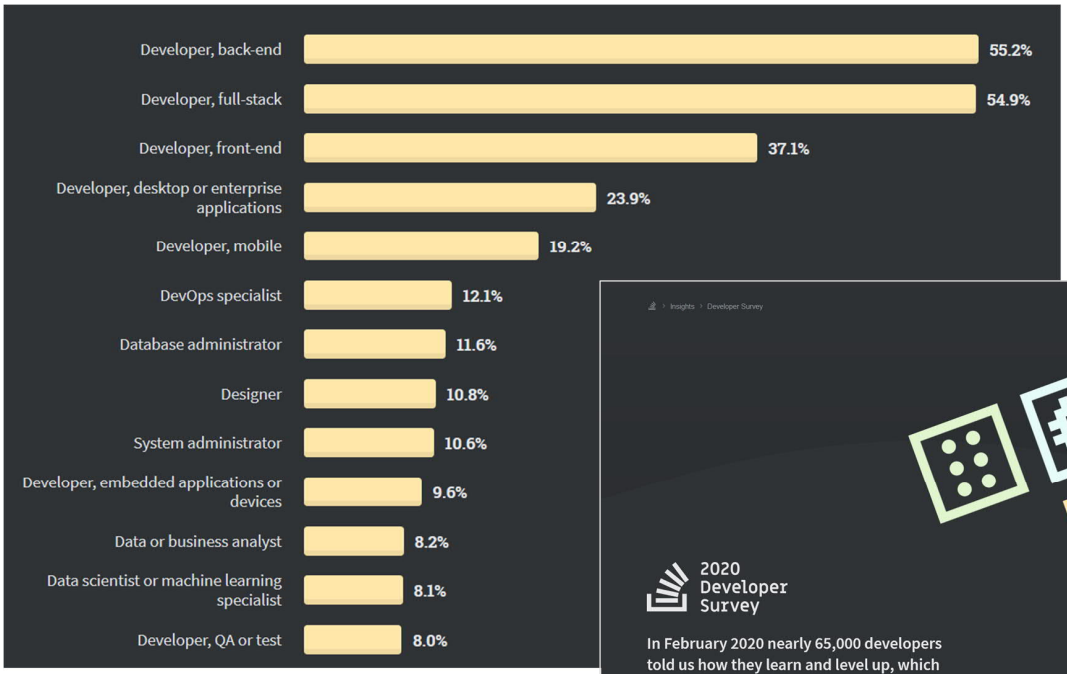
[https://www.cleart.com/what-is-cross-platform-development.html]

[https://railsware.com/blog/native-vs-hybrid-vs-cross-platform/]



Tampereen yliopisto
Tampere University

https://insights.stackoverflow.com/survey/2020



Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University

COMP.SE.100-EN (ItSE)
Introduction to Software Engneering

Lecture 6, 07.10.2020

Tensu: remember to pause Zoom lecture recording

Zoom lecture break, 10 minutes stretching, walking, etc.

Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University

Real-time systems

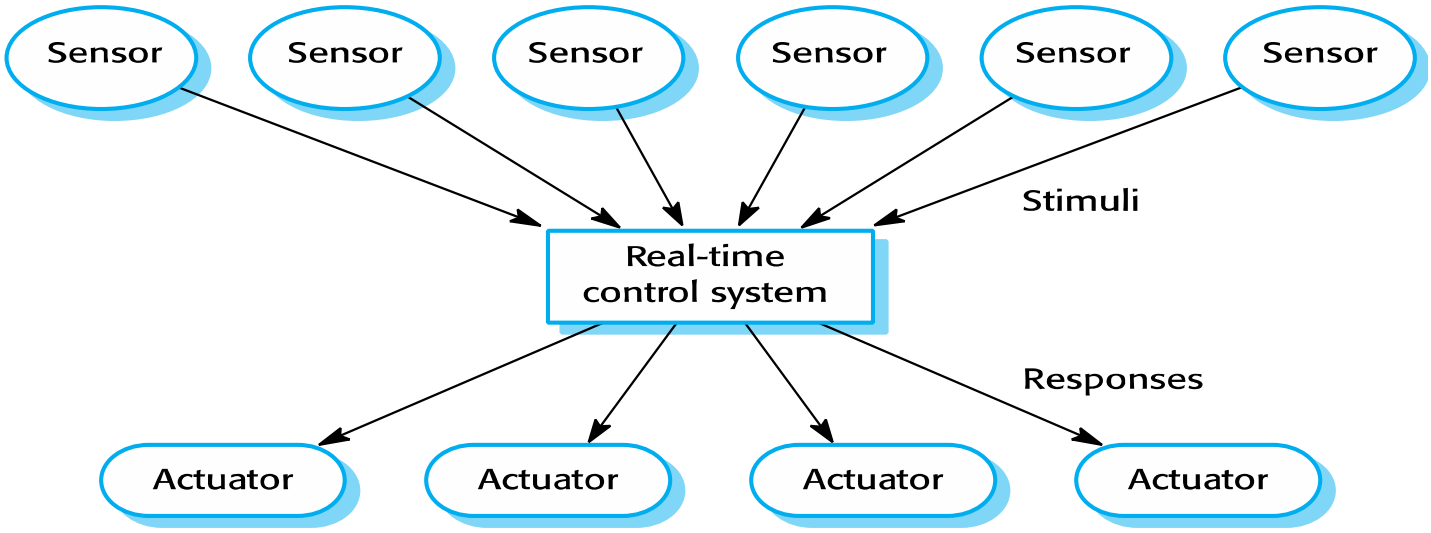
Tampereen yliopisto
Tampere University

ISO 24765:2017

3.3327 , real-time

1. problem, system, or application that is concurrent and has timing constraints whereby incoming events must be processed within a given timeframe
2. pertaining to a system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process.

A general model of an embedded real-time system



Architectural considerations

- Because of the need to respond to **timing demands** made by different stimuli/responses, the system architecture must allow for fast switching between stimulus handlers.
- Timing demands of different stimuli are different so a simple sequential loop is not usually adequate.
- Real-time systems are therefore usually designed as cooperating processes with a real-time executive controlling these processes.

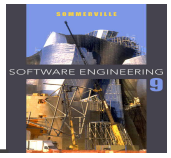
Application	Response Time Range
Speech and Audio Systems	100 ns - 10 ms
Flight Simulation	1- 10 micro sec
Robot controllers	1 ms - 10 ms
Process control systems	100 micro sec - 10 ms industrial automation
Medical Diagnosis and lab	1 ms - 100 ms automation

Table 1: Response times or deadlines of several real-time applications.

[http://community.wvu.edu/]

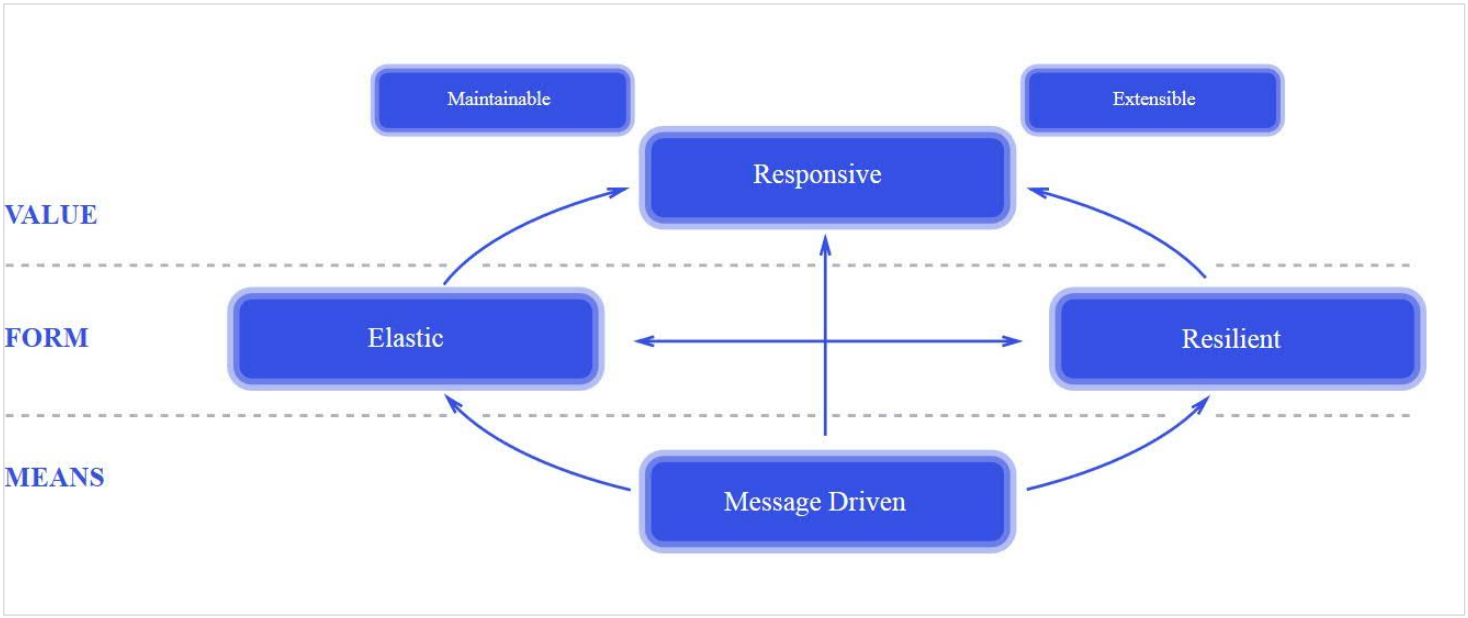
Reactive systems

Reactive systems



- Real-time systems are often considered to be reactive systems. Given a stimulus, the system must produce a reaction or response within a specified time.
- Periodic stimuli. Stimuli which occur at predictable time intervals
  - For example, a temperature sensor may be polled 10 times per second.
- Aperiodic stimuli. Stimuli which occur at unpredictable times
  - For example, a system power failure may trigger an interrupt which must be processed by the system.

https://www.reactivemanifesto.org/



https://www.reactivemanifesto.org/

Reactive Systems are (1/2):

Responsive: The system responds in a timely manner if at all possible. Responsiveness is the cornerstone of usability and utility, but more than that, responsiveness means that problems may be detected quickly and dealt with effectively. Responsive systems focus on providing rapid and consistent response times, establishing reliable upper bounds so they deliver a consistent quality of service. This consistent behaviour in turn simplifies error handling, builds end user confidence, and encourages further interaction.

Resilient: The system stays responsive in the face of failure. This applies not only to highly-available, mission-critical systems — any system that is not resilient will be unresponsive after a failure. Resilience is achieved by replication, containment, isolation and delegation. Failures are contained within each component, isolating components from each other and thereby ensuring that parts of the system can fail and recover without compromising the system as a whole. Recovery of each component is delegated to another (external) component and high-availability is ensured by replication where necessary. The client of a component is not burdened with handling its failures.

Tampereen yliopisto
Tampere University

https://www.reactivemanifesto.org/

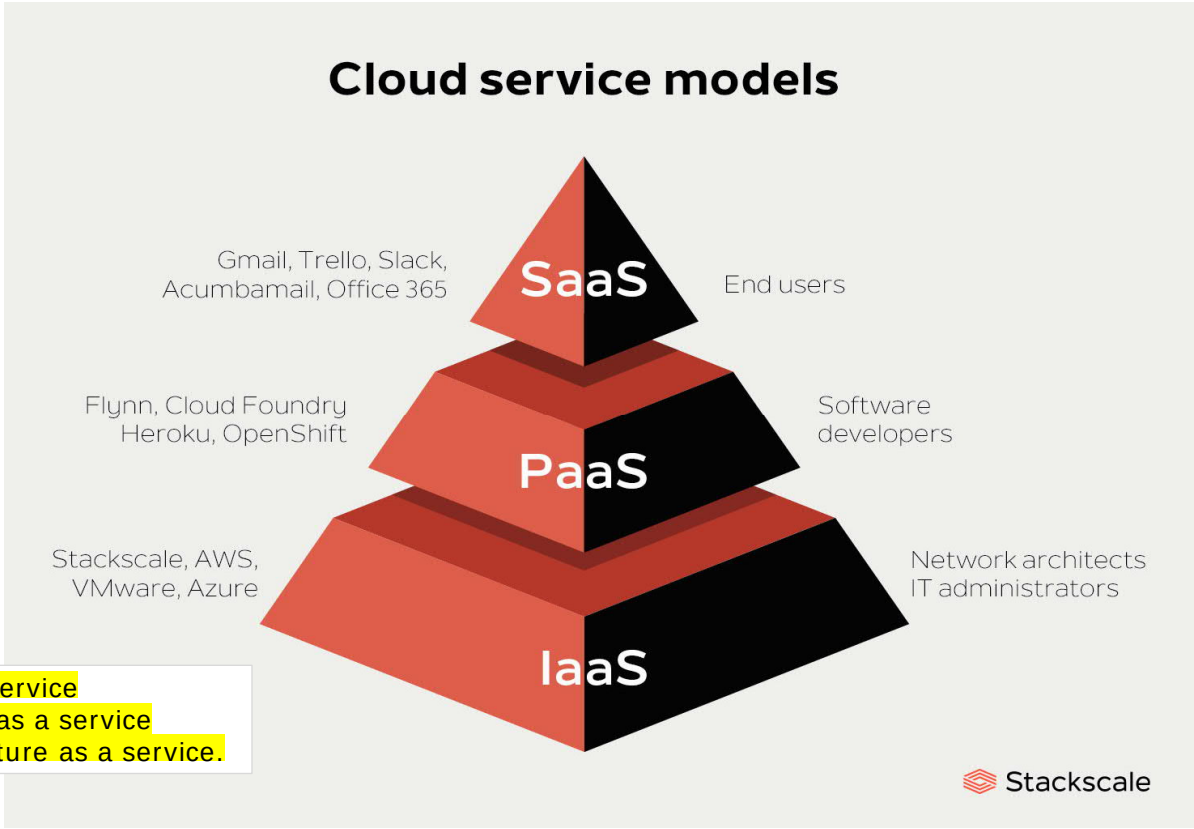
Reactive Systems are (2/2):

Elastic: The system stays responsive under varying workload. Reactive Systems can react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs. This implies designs that have no contention points or central bottlenecks, resulting in the ability to shard or replicate components and distribute inputs among them. Reactive Systems support predictive, as well as Reactive, scaling algorithms by providing relevant live performance measures. They achieve elasticity in a cost-effective way on commodity hardware and software platforms.

Message Driven: Reactive Systems rely on asynchronous message-passing to establish a boundary between components that ensures loose coupling, isolation and location transparency. This boundary also provides the means to delegate failures as messages. Employing explicit message-passing enables load management, elasticity, and flow control by shaping and monitoring the message queues in the system and applying back-pressure when necessary. Location transparent messaging as a means of communication makes it possible for the management of failure to work with the same constructs and semantics across a cluster or within a single host. Non-blocking communication allows recipients to only consume resources while active, leading to less system overhead.

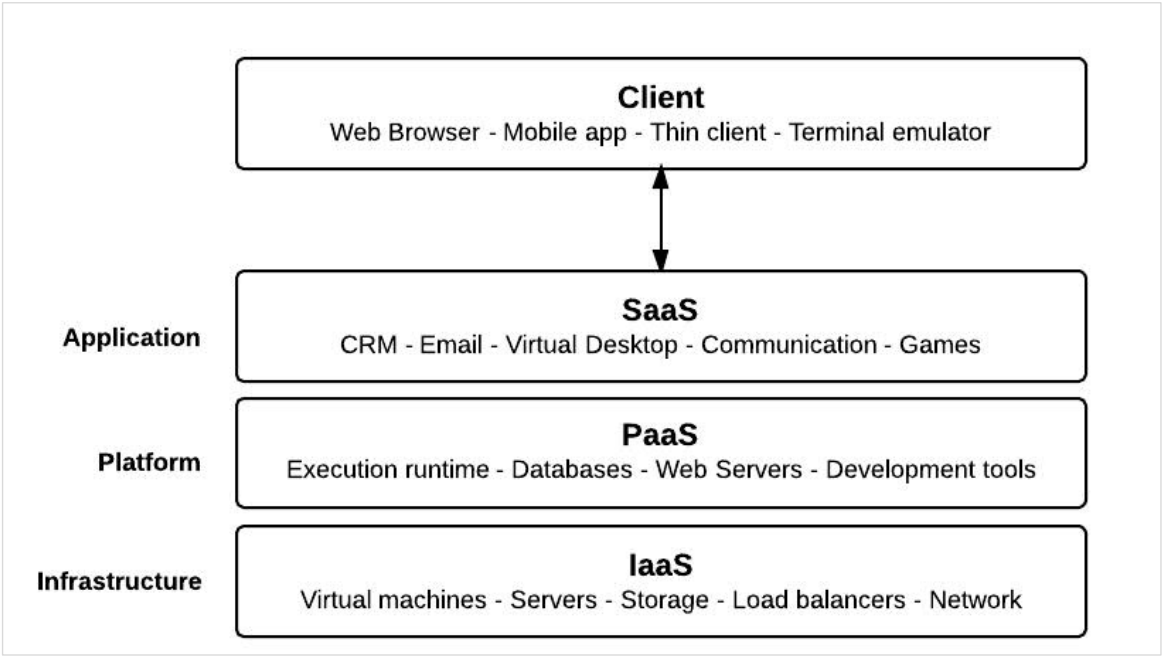
Tampereen yliopisto
Tampere University

SaaS, PaaS, IaaS,...
Cloud computing



SaaS = sw as a service
PaaS = platform as a service
IaaS = infrastructure as a service.

Different Types of Cloud Computing Service Models



[https://www.bluepiit.com/blog/different-types-of-cloud-computing-service-models/]

Cloud computing

There are several types of cloud computing. Providers offer infrastructure, platforms or software as a web-based service. Instead of purchasing these components, clients subscribe to them as a variable-cost service. The service provider owns the components and is responsible for housing, running and maintaining them.

Infrastructure as a Service (IaaS)

- the service provider offers computer infrastructure components as a web-based service
- for example servers, storage, data centre space and network equipment

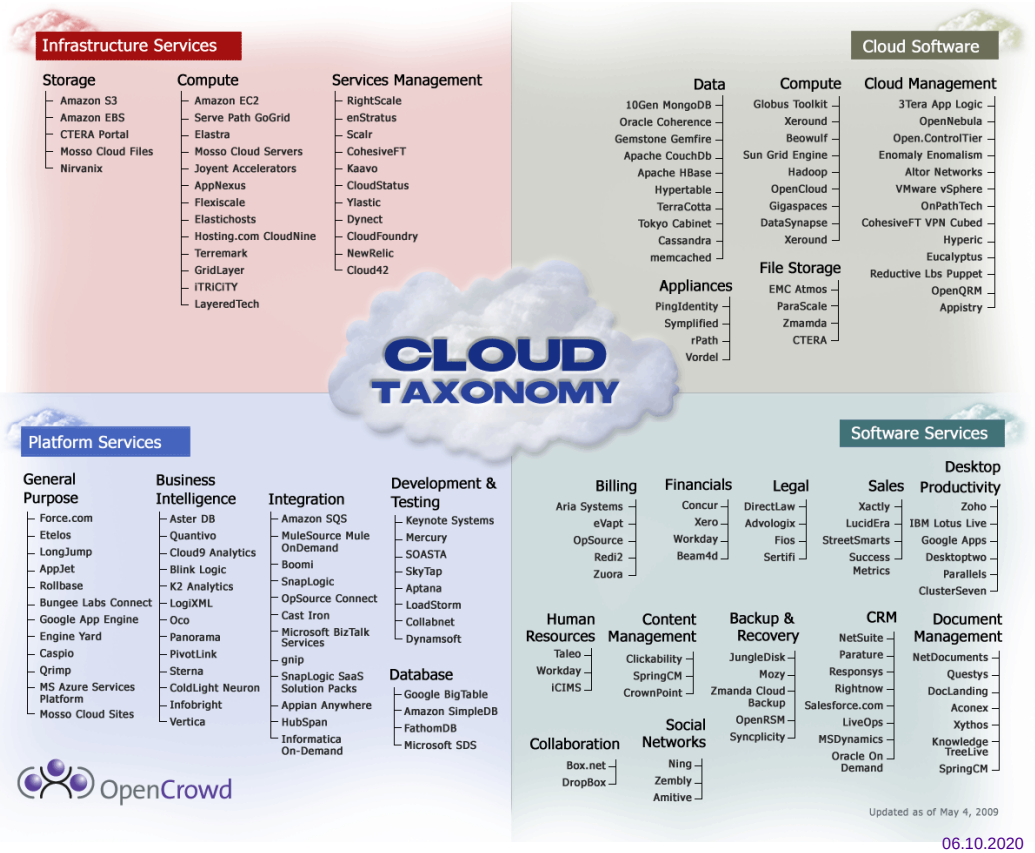
Platform as a Service (PaaS)

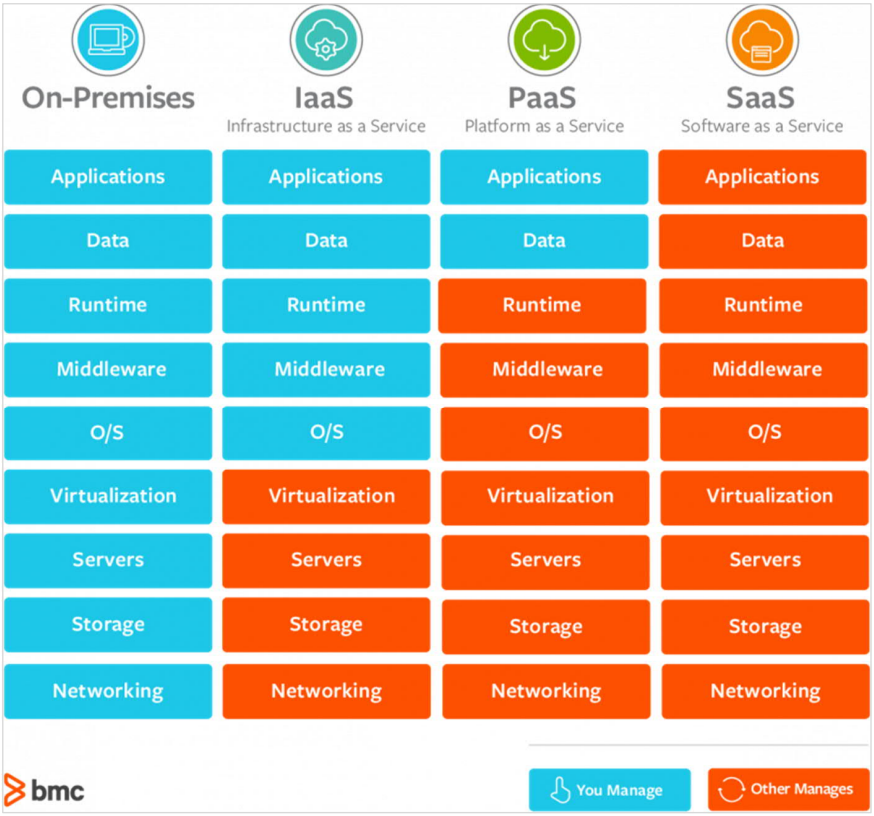
- the service provider offers a computing platform as a web-based service
- this typically includes an operating system, programming language, execution environment, database and web server

Software as a Service (SaaS)

- the service provider offers software applications as a web-based service
- for example horizontal business applications like Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Content Management (CM), Human Resource Management (HRM) or finance and accounting.

[https://www.cbi.eu/market-information/outsourcing-bpo-ito/cloud-computing/europe/]





SaaS, 1

Stands for "Software as a Service." SaaS is software that is deployed over the Internet rather than installed on a computer. It is often used for enterprise applications that are distributed to multiple users. SaaS applications typically run within a Web browser, which means users only need a compatible browser in order to access the software.

SaaS is considered part of cloud computing since the software is hosted on the Internet, or the "cloud". Because SaaS applications are accessed from a remote server rather than installed on individual machines, it is easy to maintain the software for multiple users. For example, when the remote software is updated, the client interface is also updated for all users. This eliminates incompatibilities between different software versions and allows vendors to make incremental updates without requiring software downloads. Additionally, users can save data to a central online location, which makes it easy to share files and collaborate on projects.

Tampereen yliopisto
Tampere University

SaaS, 2 [www.bmc.com]

Software as a Service, also known as cloud application services.

There are a few ways to help you determine when SaaS is being utilized:

- Managed from a central location
- Hosted on a remote server
- Accessible over the internet
- Users not responsible for hardware or software updates.

SaaS may be the most beneficial option in several situations, including:

- Startups or small companies that need to launch ecommerce quickly and don't have time for server issues or software
- Short-term projects that require quick, easy, and affordable collaboration
- Applications that aren't needed too often, such as tax software
- Applications that need both web and mobile access.

Tampereen yliopisto
Tampere University

SaaS, 3 [www.bmc.com]

Software as a Service, also known as cloud application services.

SaaS Limitations and Concerns

- Interoperability.
- Vendor lock-in.
- Lack of integration support.
- Data security.
- Customization.
- Lack of control.
- Feature limitations.
- Performance and downtime.

Tampereen yliopisto
Tampere University

IaaS, 1 [www.bmc.com]

Cloud infrastructure services, known as Infrastructure as a Service (IaaS), are made of highly scalable and automated compute resources. IaaS is fully self-service for accessing and monitoring computers, networking, storage, and other services. IaaS allows businesses to purchase resources on-demand and as-needed instead of having to buy hardware outright.

IaaS offers many advantages, including:

- The most flexible cloud computing model
- Easy to automate deployment of storage, networking, servers, and processing power
- Hardware purchases can be based on consumption
- Clients retain complete control of their infrastructure
- Resources can be purchased as-needed
- Highly scalable.

Tampereen yliopisto
Tampere University

IaaS, 2 [www.bmc.com]

Characteristics that define IaaS include:

- Resources are available as a service
- Cost varies depending on consumption
- Services are highly scalable
- Multiple users on a single piece of hardware
- Organization retain complete control of the infrastructure
- Dynamic and flexible.

IaaS Limitations and Concerns, particular limitations to IaaS include:

- Security.
- Legacy systems operating in the cloud.
- Internal resources and training.
- Multi-tenant security.

Tampereen yliopisto
Tampere University

PaaS, 1 [www.bmc.com]

Cloud platform services, also known as Platform as a Service (PaaS), provide cloud components to certain software while being used mainly for applications. PaaS delivers a framework for developers that they can build upon and use to create customized applications. All servers, storage, and networking can be managed by the enterprise or a third-party provider while the developers can maintain management of the applications.

PaaS offers numerous advantages, including:

- Simple, cost-effective development and deployment of apps
- Scalable
- Highly available
- Developers can customize apps without the headache of maintaining the software
- Significant reduction in the amount of coding needed
- Automation of business policy
- Easy migration to the hybrid model.

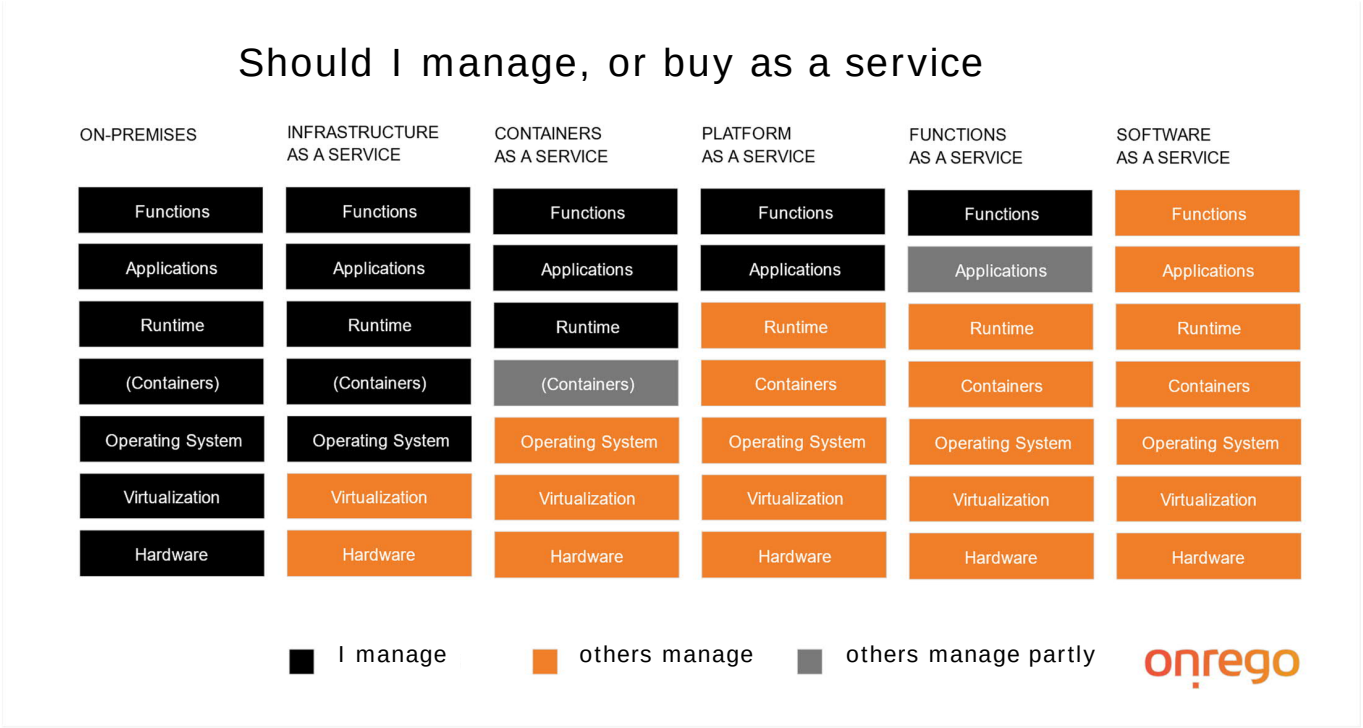
Tampereen yliopisto
Tampere University

PaaS, 2 [www.bmc.com]

PaaS Limitations and Concerns

- Data security.
- Integrations.
- Vendor lock-in.
- Customization of legacy systems.
- Runtime issues.
- Operational limitation.

XaaS...



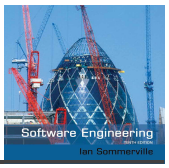
Tampereen yliopisto
Tampere University

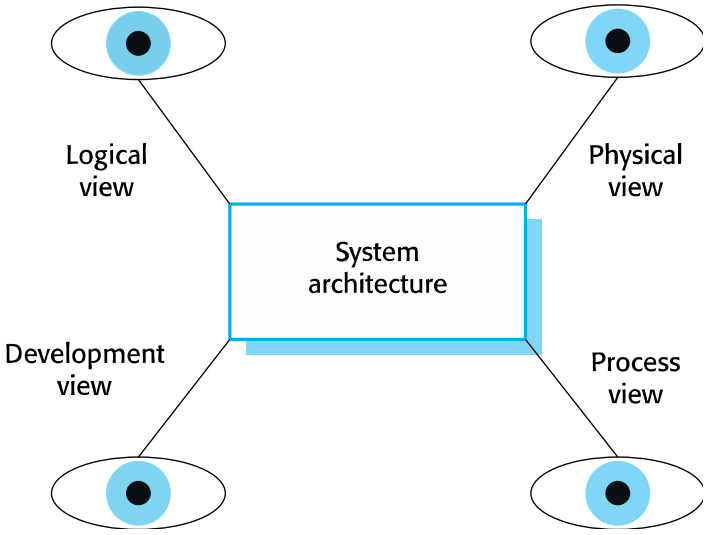
Software systems Architectures

- Software systems
- Architectures

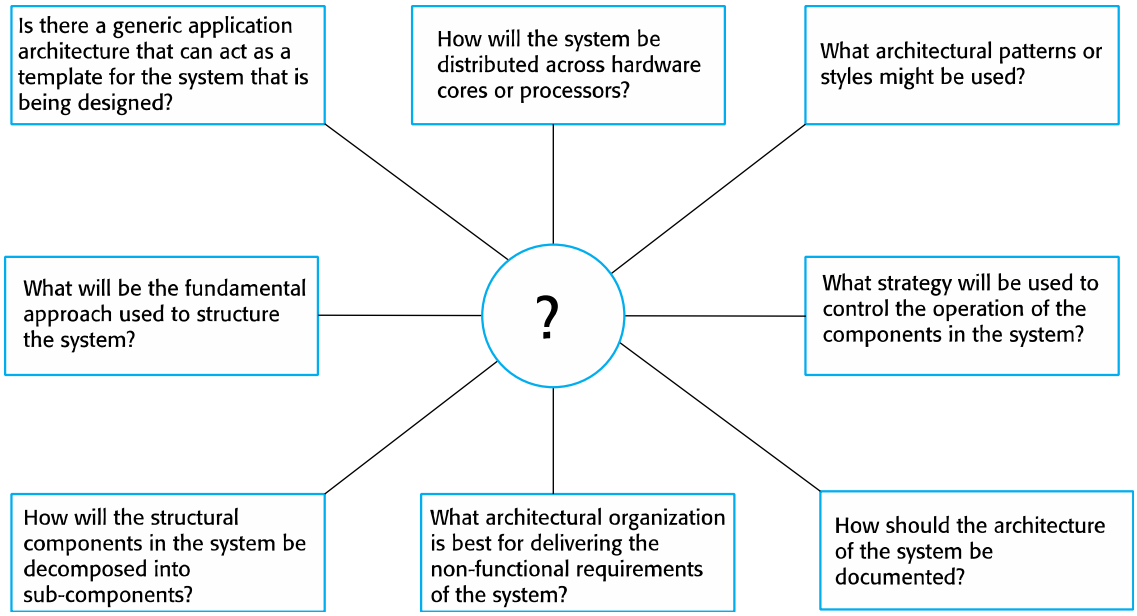
(it is difficult, if not impossible, to define one clear exact definition for those...)

Architectural views





Architectural design decisions

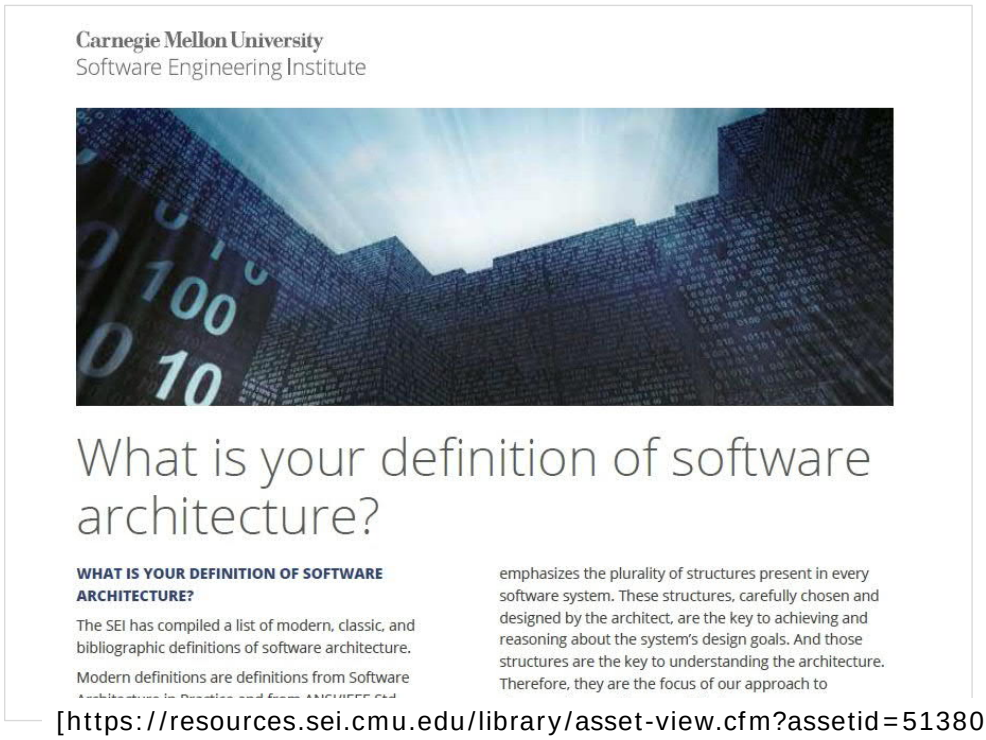


Tampereen yliopisto
Tampere University

Architecture Framework

Definition of Software Architecture... it depends...

CMU/SEI



[https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=513807], 2017

Architectural views [Sommerville]

It is impossible to represent all relevant information about a system's architecture in a single diagram, as a graphical model can only show one view or perspective of the system. It might show how a system is decomposed into modules, how the runtime processes interact, or the different ways in which system components are distributed across a network. Because all of these are useful at different times, for both design and documentation, you usually need to present multiple views of the software architecture.

There are different opinions as to what views are required.

1. A logical view, which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.

2. A process view, which shows how, at runtime, the system is composed of interacting processes. This view is useful for making judgments about non-functional system characteristics such as performance and availability.

3. A development view, which shows how the software is decomposed for development; that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful for software managers and programmers.

4. A physical view, which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.

5. A Conceptual view.

Tampereen yliopisto
Tampere University

Architecture vs. framework, 1

Framework is an implementation of an architecture.

- - -

An architecture is the the abstract design concept of an application. Basically, a structure of the moving parts and how they're connected. A framework is a pre-built general or special purpose architecture that's designed to be extended.

- - -

If an architecture is the design of a structure, a framework is the architecture of a foundation. Frameworks are specifically designed to be built on or extended.

- - -

[softwareengineering.stackexchange.com]

Tampereen yliopisto
Tampere University

Architecture vs. framework, 2

Generally speaking, architecture is an abstract plan that can include design patterns, modules, and their interactions. Frameworks are architected "physical" structures on which you build your application.

Your architecture may incorporate multiple frameworks. And multiple layers of framework. At each layer, the "architecture" is the often-documented "thinking" from which the layer is built. The frameworks are the "physical" components used to build it.

- - -

In software, a Framework is a software module or set of modules that supports a generic programming concept by abstracting common functionality (code in a software sense) into a reusable format.

An Architecture is an assembly of systems that solves business needs.

- - -

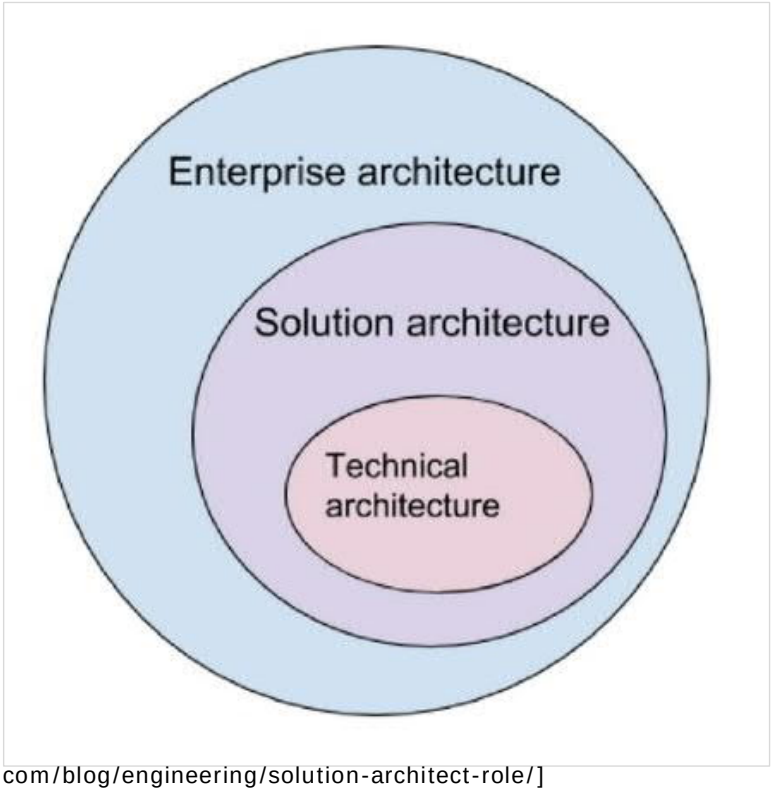
[softwareengineering.stackexchange.com]

Tampereen yliopisto
Tampere University

Enterprise architect focuses on building complex enterprise ecosystems and solves high-level strategic issues. Enterprise architecture defines strategic directions of the business architecture, which then leads to an understanding of what technology facilities are needed to support that architecture.

Solution architecture (SA) is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.

Technical architect is mainly in charge of engineering problems and software architecture.



[https://www.altexsoft.com/blog/engineering/solution-architect-role/]

Tampereen yliopisto
Tampere University

Just another classification of architecture layers

Enterprise architecture (FI: kokonaisarkkitehtuuri, JHS 179-2017)
”company-wide total architecture”

Software architecture (FI: ohjelmistoarkkitehtuuri)

Systems architecture (FI: järjestelmäarkkitehtuuri).

“An architecture is the result of a set of business and technical decisions.”

In architecture you should think about security and future extensions (APIs).

Tampereen yliopisto
Tampere University

About architecture, 1

Having a software architecture is important to the successful development of a software system.

Software systems are constructed to satisfy organizations' business goals. The architecture is a bridge between those (often abstract) business goals and the final (concrete) resulting system. While the path from abstract goals to concrete systems can be complex, the good news is that software architectures can be designed, analyzed, documented, and implemented using known techniques that will support the achievement of these business and mission goals. The complexity can be tamed, made tractable.

[Len Bass et al.: Software Architecture in Practice, 3rd ed., 2013]

Tampereen yliopisto
Tampere University

About architecture, 2

Architecture is defined by the recommended practice as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. This definition is intended to encompass a variety of uses of the term architecture by recognizing their underlying common elements. Principal among these is the need to understand and control those elements of system design that capture the system's utility, cost, and risk. In some cases, these elements are the physical components of the system and their relationships. In other cases, these elements are not physical, but instead, logical components. In still other cases, these elements are enduring principles or patterns that create enduring organizational structures.

[ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems]

TOGAF

- ✧ The TOGAF framework has been developed by the Open Group as an open standard and is intended to support the design of a business architecture, a data architecture, an application architecture and a technology architecture for an enterprise.
- ✧ At its heart is the Architecture Development Method (ADM), which consists of a number of discrete phases.

Tampereen yliopisto
Tampere University

TOGAF = The Open Group Architecture Framework

The TOGAF® Standard — Version 9.2

Structure of the TOGAF® Standard

THE Open GROUP

The TOGAF Standard, Version 9.2, is divided into six sections as follows:

PART I: Introduction
A high-level introduction to the key concepts of Enterprise Architecture and in particular the TOGAF approach. It contains the definitions of terms used throughout the standard.

PART II: Architecture Development Method
The core of the TOGAF framework. It describes the TOGAF Architecture Development Method (ADM) – a step-by-step approach to developing an Enterprise Architecture.

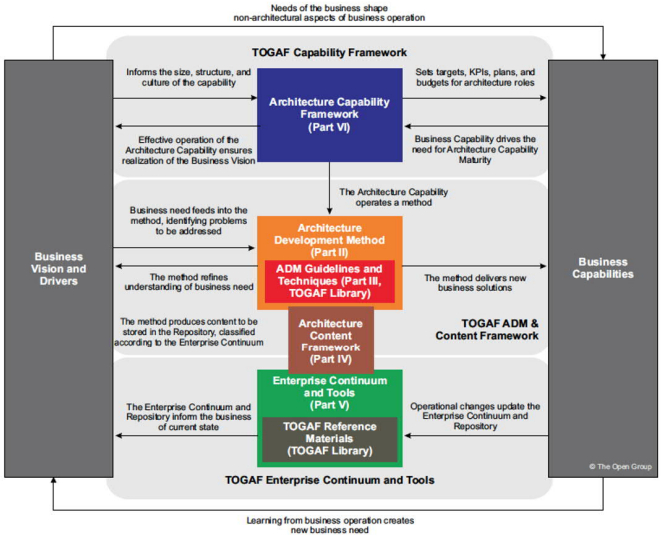
PART III: ADM Guidelines and Techniques
A collection of guidelines and techniques available for use in applying the TOGAF approach and the TOGAF ADM. Additional guidelines and techniques are also in the TOGAF Library.

PART IV: Architecture Content Framework
The TOGAF content framework, including a structured metamodel for architectural artifacts, the use of re-usable architecture building blocks, and an overview of typical architecture deliverables.

PART V: Enterprise Continuum & Tools
Taxonomies and tools to categorize and store the outputs of architecture activity within an enterprise.

PART VI: Architecture Capability Framework
The organization, processes, skills, roles, and responsibilities required to establish and operate an architecture function within an enterprise.

Accompanying the standard is the TOGAF Library. The TOGAF Library is a reference library containing guidelines, templates, patterns, and other forms of reference material to accelerate the creation of new architectures for the enterprise.



N180 Reference Cards: The TOGAF Standard, Version 9.2 Overview

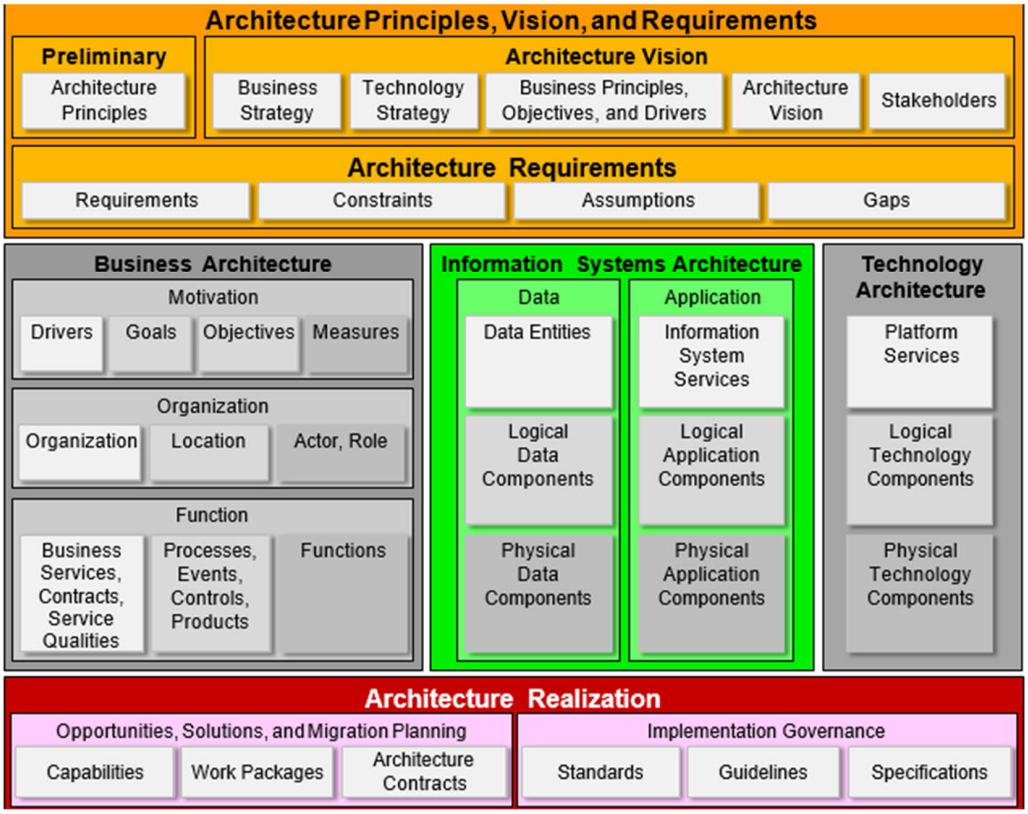
Copyright © 2018 The Open Group. All Rights Reserved.
TOGAF® is a registered trademark of The Open Group.

Tampereen yliopisto
Tampere University

TOGAF 9

TOGAF =
The Open Group Architecture Framework

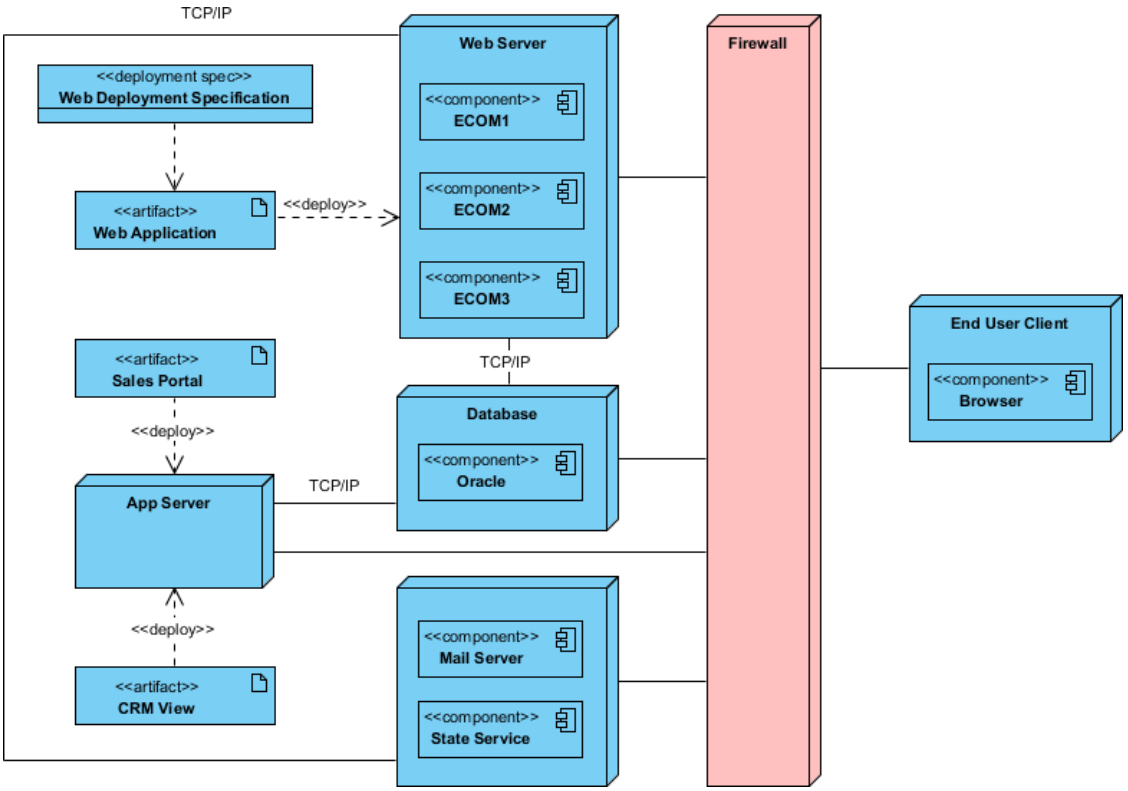
(FI: kokonais-arkkitehtuuri-viitekehys)



Tampereen yliopisto
Tampere University

Client-server

APIs (interfaces, connections) are taken into account, i.e. planned at design phase.



[https://www.archimetric.com/what-is-deployment-diagram/]

Some architecture standards

- ISO/IEC/ IEEE 42010:2011

Systems and software engineering — Architecture description

- IEEE 42020:2019

ISO/IEC/IEEE International Standard - Software, systems and enterprise -- Architecture processes

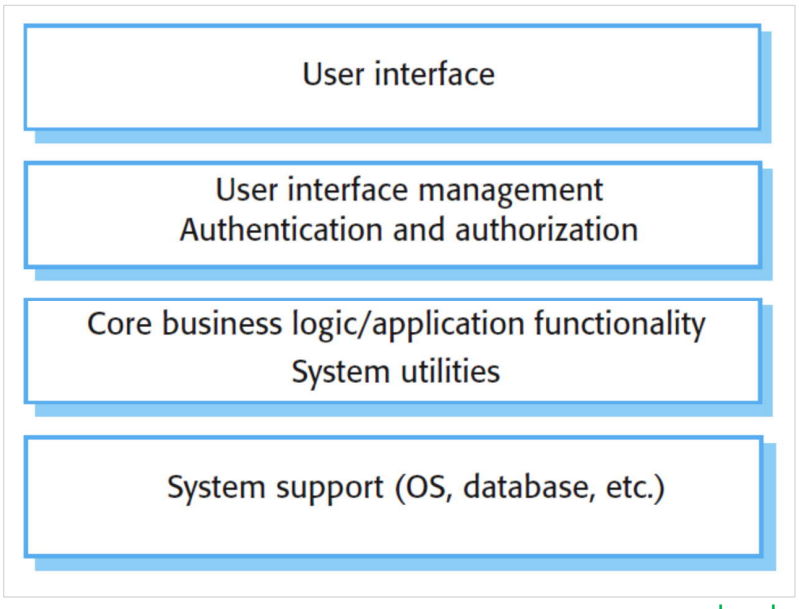
- ISO/IEC/IEEE 42030:2019

Software, systems and enterprise — Architecture evaluation framework

"An architecture is what is fundamental to a system — not necessarily everything about a system, but the essentials."

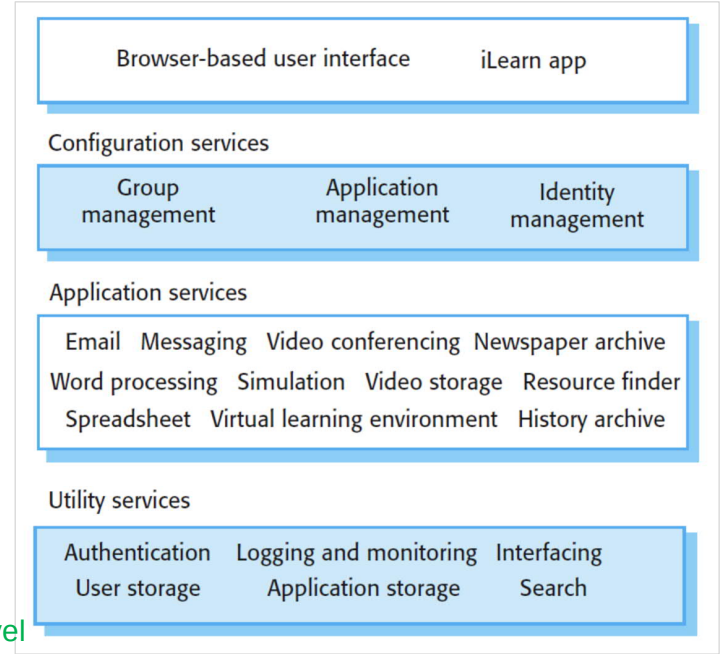
One example of architecture layers [Sommerville, 2016]

high level



Generic layered architecture.

low level



iLearn system architecture

Architectural views [Sommerville]

It is impossible to represent all relevant information about a system's architecture in a single diagram, as a graphical model can only show one view or perspective of the system. It might show how a system is decomposed into modules, how the runtime processes interact, or the different ways in which system components are distributed across a network. Because all of these are useful at different times, for both design and documentation, you usually need to present multiple views of the software architecture.

There are different opinions as to what views are required.

1. A logical view, which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.

2. A process view, which shows how, at runtime, the system is composed of interacting processes. This view is useful for making judgments about non-functional system characteristics such as performance and availability.

3. A development view, which shows how the software is decomposed for development; that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful for software managers and programmers.

4. A physical view, which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.

5. A Conceptual view.

Architecture vs. framework, 1

Framework is an implementation of an architecture.

- - -

An architecture is the the abstract design concept of an application. Basically, a structure of the moving parts and how they're connected. A framework is a pre-built general or special purpose architecture that's designed to be extended.

- - -

If an architecture is the design of a structure, a framework is the architecture of a foundation. Frameworks are specifically designed to be built on or extended.

- - -

[softwareengineering.stackexchange.com]

Tampereen yliopisto
Tampere University

Architecture vs. framework, 2

Generally speaking, architecture is an abstract plan that can include design patterns, modules, and their interactions. Frameworks are architected "physical" structures on which you build your application.

Your architecture may incorporate multiple frameworks. And multiple layers of framework. At each layer, the "architecture" is the often-documented "thinking" from which the layer is built. The frameworks are the "physical" components used to build it.

- - -

In software, a Framework is a software module or set of modules that supports a generic programming concept by abstracting common functionality (code in a software sense) into a reusable format.

An Architecture is an assembly of systems that solves business needs.

- - -

[softwareengineering.stackexchange.com]

Tampereen yliopisto
Tampere University

Terminology, ISO/IEC/IEEE 24765:2017 standard

3.169 , application architecture

1. architecture including the architectural structure and rules (e.g. common rules and constraints) that constrains a specific member product within a product line

Note 1 to entry: The application architecture captures the high-level design of a specific member product of a product line.

3.210 , architecting

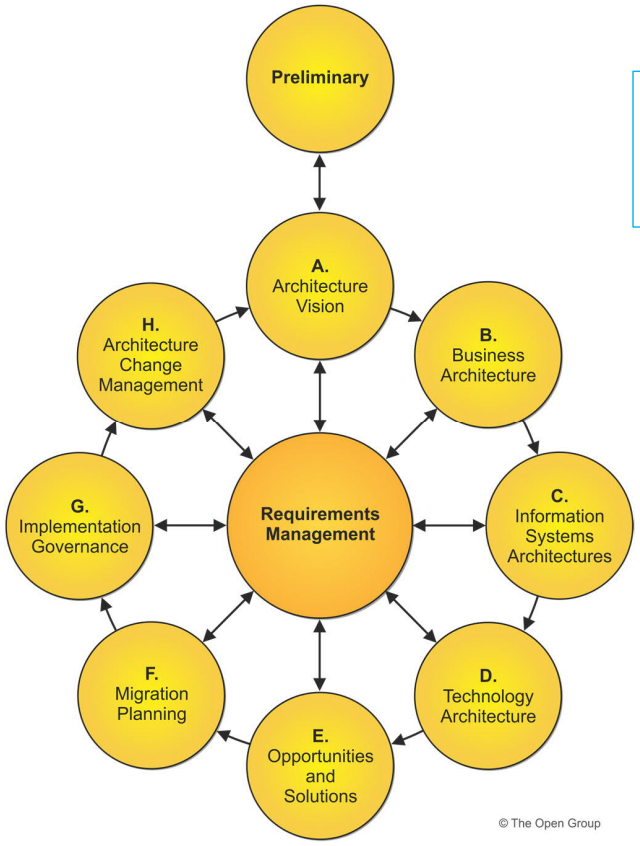
1. process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system's life cycle.

3.214 , architectural structure

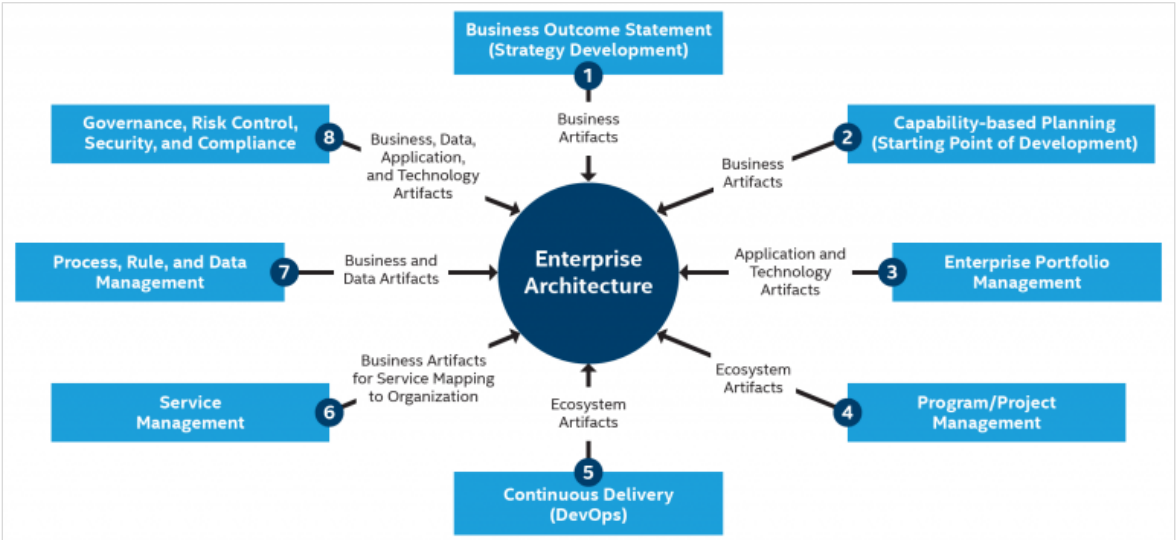
1. physical or logical layout of the components of a system design and their internal and external connections.

TOGAF (The Open Group Architecture Framework) 9.2 architecture

Planning enterprise architecture requires balancing a wide variety of governance, technology and management issues.



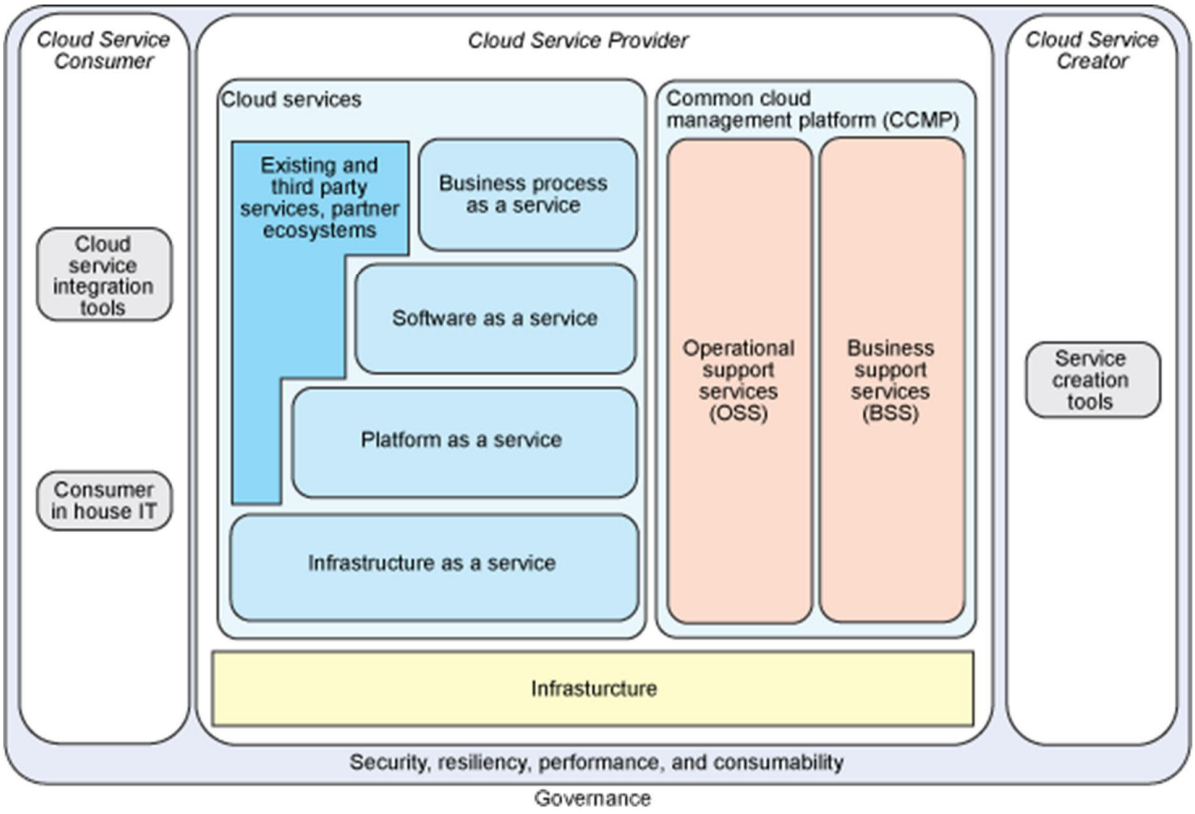
TOGAF ADM (Architecture Development Model)



” Enterprise architects have a tough job. They have to think strategically but act tactically. A successful enterprise architect can sit down at the boardroom table and discuss where the business needs to go, then translate “business speak” into technical capabilities on the back end. The key to EA is to always focus on business needs first, then how those needs can be met by applying technology.”

[https://itpeernetwork.intel.com/enterprise-architecture/]

IBM Cloud Architecture



Example of Zachman's 6x6 matrix,

top and left sides are main views.

	WHAT	HOW	WHERE	WHO	WHEN	WHY	
SCOPE CONTEXTS	Inventory Identification / Inventory Types	Process Identification / Process Types	Network Identification / Network Types	Organization Identification / Organization Types	Timing Identification / Timing Types	Motivation Identification / Motivation Types	STRATEGISTS AS THEORISTS
BUSINESS CONCEPTS	Inventory Definition / Business Entity Business Relationship	Process Definition / Business Transform Business Input	Network Definition / Business Location Business Connection	Organization Definition / Business Role Business Work	Timing Definition / Business Cycle Business Moment	Motivation Definition / Business End Business Means	EXECUTIVE LEADERS AS OWNERS
SYSTEM LOGIC	Inventory Representation / System Entity System Relationship	Process Representation / System Transform System Input	Network Representation / System Location System Connection	Organization Representation / System Role System Work	Timing Representation / System Cycle System Moment	Motivation Representation / System End System Means	ARCHITECTS AS DESIGNERS
TECHNOLOGY PHYSICS	Inventory Specification / Technology Entity Technology Relationship	Process Specification / Technology Transform Technology Input	Network Specification / Technology Location Technology Connection	Organization Specification / Technology Role Technology Work	Timing Specification / Technology Cycle Technology Moment	Motivation Specification / Technology End Technology Means	ENGINEERS AS BUILDERS
COMPONENT ASSEMBLIES	Inventory Configuration / Component Entity Component Relationship	Process Configuration / Component Transform Component Input	Network Configuration / Component Location Component Connection	Organization Configuration / Component Role Component Work	Timing Configuration / Component Cycle Component Moment	Motivation Configuration / Component End Component Means	TECHNICIANS AS IMPLEMENTERS
OPERATIONS CLASSES	Inventory Instantiation / Operations Entity Operations Relationship	Process Instantiation / Operations Transform Operations Input	Network Instantiation / Operations Location Operations Connection	Organization Instantiation / Operations Role Operations Work	Timing Instantiation / Operations Cycle Operations Moment	Motivation Instantiation / Operations End Operations Means	WORKERS AS PARTICIPANTS
	INVENTORY SETS	PROCESS TRANSFORMATIONS	NETWORK NODES	ORGANIZATION GROUPS	TIMING PERIODS	MOTIVATION REASONS	

[https://cdn.visual-paradigm.com/guide/enterprise-architecture/what-is-zachman-framework/01-zachman-framework.png]

just another

Zachman 6x6

	Data (**What**)	Function (**How**)	Network (**Where**)	People (**Who**)	Time (**When**)	Motivation (**Why**)
Objectives / Scope	List of things important to the enterprise	List of processes the enterprise performs	List of locations where the enterprise operates	List of organizational units	List of business events / cycles	List of business goals / strategies
Business Model	Entity relationship diagram (including m:m, n-ary, attributed relationships)	Business process model (physical data flow diagram)	Logistics network (nodes and links)	Organization chart, with roles; skill sets; security issues.	Business master schedule	Business plan
Information System Model	Data model (converged entities, fully normalized)	Essential Data flow diagram; application architecture	Distributed system architecture	Human interface architecture (roles, data, access)	Dependency diagram, entity life history (process structure)	Business rule model
Technology Model	Data architecture (tables and columns); map to legacy data	System design: structure chart, pseudo-code	System architecture (hardware, software types)	User interface (how the system will behave); security design	"Control flow" diagram (control structure)	Business rule design
Detailed Representation	Data design (denormalized), physical storage design	Detailed Program Design	Network architecture	Screens, security architecture (who can see what?)	Timing definitions	Rule specification in program logic
Function System	Converted data	Executable programs	Communications facilities	Trained people	Business events	Enforced rules

open system architecture

Vendor-independent, non-proprietary, computer system or device design based on official and/or popular standards. It allows all vendors (in competition with one another) to create add-on products that increase a system's (or device's) flexibility, functionality, interoperatability, potential use, and useful life. And enables the users to customize and extend a system's (or device's) capabilities to suit individual requirements. Also called open architecture.

[http://www.businessdictionary.com/definition/open-system-architecture.html]

Highlights - What to remember

- software systems can be classified in many many ways
- there are also many architectural views, by which you can describe and define a business and software system
- framework is one realisation of an architecture
- At requirements and design phase when defining the architecture and framework, besides business needs, you have better to take possible future extensions into account, e.g. interfaces (API = application programming interface) to other systems or future extensions. Those are easier to plan right from the beginning, than add later.
- cross-paltform development tools are not silver bullets
- when building a software system, think where you have the data, and where you process it, so what is your actual need for the ICT system (product or service).

Now the additional L6 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Now the additional L6 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Now the additional L6 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Now the additional L6 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

popularity trend of programming languages

Language	Origin	2015	2016	2017	2018	2019	Main Purpose
Javascript	1995	54.4%	55.4%	62.5%	71.5%	67.8%	Web development, Dynamic contents, Client and Server side
Java	1995	37.4%	36.3%	39.7%	45.4%	41.1%	Enterprise Application
Bash/Shell	1971/79	–	–	–	40.4%	36.6%	Automation and system admin
Python	1991	23.8%	24.9%	32.0%	37.9%	41.7%	General purpose
PHP	1995	29.7%	25.9%	28.1%	31.4%	26.4%	Web development, Server-side
C++	1980/83	20.6%	19.4%	22.3%	24.6%	23.5%	General purpose
C	1972	16.4%	15.5%	19.0%	22.1%	20.6%	General Purpose, Low-level

CODING INFINITE
Your Infinite Coding Solutions

ANDROID JAVA .NET CORE PHP FORUM TOP LANGUAGES POLL

JOB MARKET AND SALARY

Salary depends upon the geographical area and demand of the products, a programming language based salary comparison is just a tool to predict or estimate the salary trend. We have summarized salary based on programming language from popular survey i.e. Dice salary survey 2018 and Stack-overflow survey 2018 and 2019.

Programming Language	The Dice salary survey (2018)	Stack-overflow Survey (2018)	Stack-overflow Survey (2019)
Go	$132,827	$66K	$80K
Perl	$110,678	$69K	NA
Shell	$109,518	$63K	$69K
Node.js JavaScript	$105,418	$55K	$56K
Java/J2EE	$105,164	NA	$52K
TypeScript	$103,680	$60K	$60K

Tampereen yliopisto
Tampere University

Classification of software systems

Tampereen yliopisto
Tampere University

Just another software classification

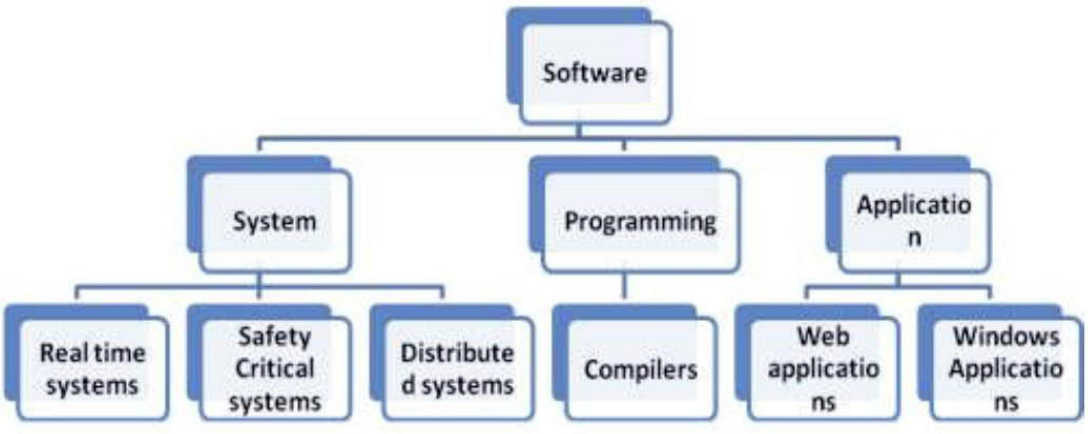
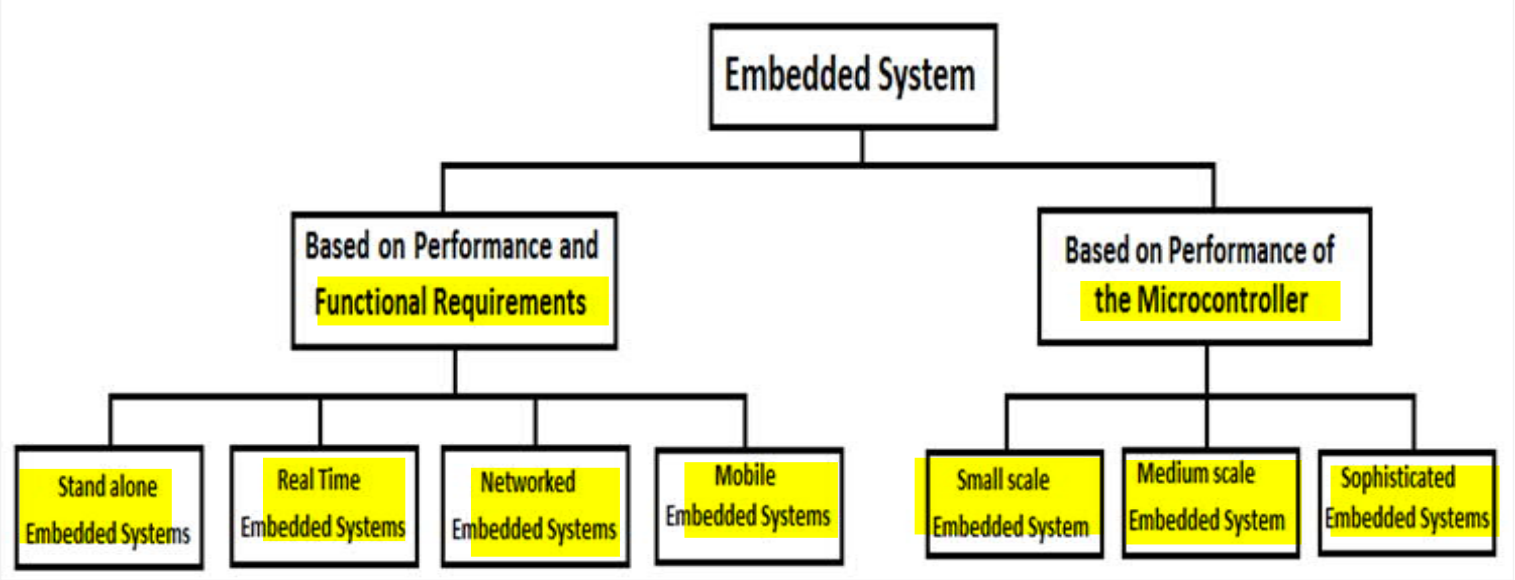


Figure 1. Software Classification

[https://www.researchgate.net/figure/Software-Classification_fig5_288645078]

One classification of embedded systems



[https://er.yuvayana.org/classification-of-embedded-system-with-details/]

Software Systems Taxonomy

(As found in Alleman's "Agile Project Management Methods for IT Projects")

Software Type	Attributes
Management Information Systems	Software that an enterprise uses to support business and administrative operations.
Outsourced Systems	Software projects built for client organizations.
Systems Software	Software that controls a physical device such as a computer or a telephone switch.
Commercial Software	Software applications that are marketed to hundreds or even millions of clients.
Military Software	Software produced for the uniformed services.
End User Software	Small applications written for personal use.
Web Application and e- Projects	Small, medium, and large scale projects with legacy system integration, transaction processing, multimedia delivery, and web browser based user interfaces.

Tampereen yliopisto
Tampere University

Edge computing

Tampereen yliopisto
Tampere University

PWA
Progressive web application

Cross-platform development (mobile)

What Is Cross Platform Development, 3

Cross platform development provides the flexibility to build your app using a universal language like Javascript which can then be exported to various smartphone platforms. This allows one “app” to work across multiple habitats. This can be done in two ways:

Native App – Cross Platform Development – Using tools like React Native, Xamarin, or NativeScript, you end up with an app that still uses native APIs which allows for excellent performance across all platforms, without having to code each separately. This results in a very nice app that runs well and is a great compromise over the time and expense of building a native app.

Hybrid Apps – Cross Platform Development – Hybrid development is a tier lower than native cross platform development. It relies on a built in web browser and HTML 5, CSS, and Javascript to render the app. The app is coded and then rendered on the smartphone inside an app container which is driven by an internal web browser. In hybrid apps, developers program using native web languages but wrap it in a native wrapper for each operating system, thus rendering it usable across multiple platforms. This can work OK for simple apps but in more complex apps, performance could be an issue.

[https://www.cleart.com/what-is-cross-platform-development.html]

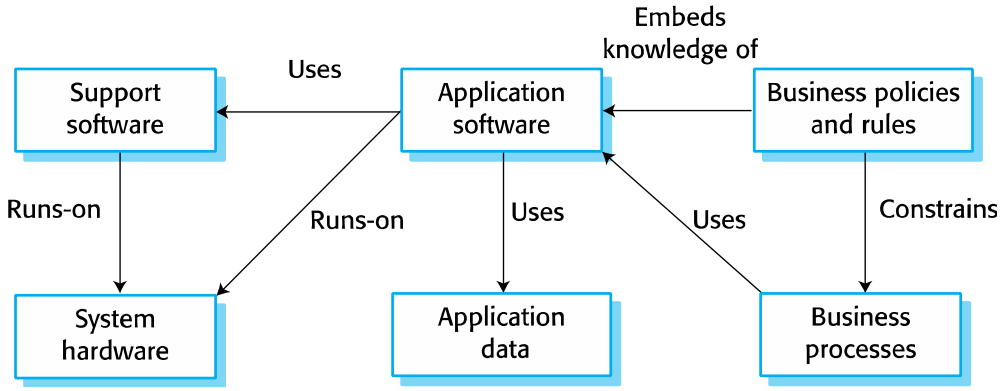
Legacy systems

Legacy systems

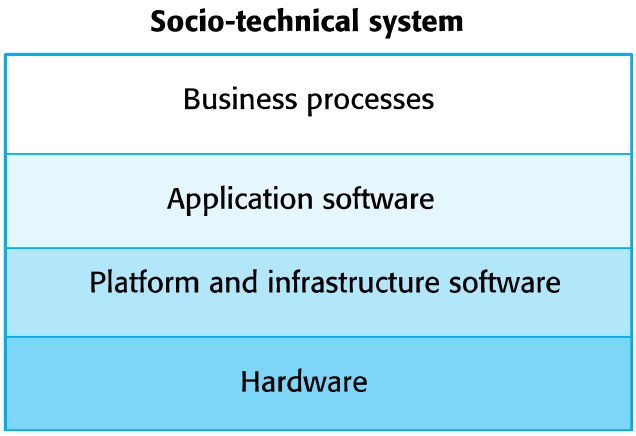
- Legacy systems are **older systems that rely on languages and technology** that are no longer used for new systems development.
- Legacy software may be dependent on older hardware, such as mainframe computers and may have associated legacy processes and procedures.
- Legacy systems are not just software systems but are broader socio-technical systems that include hardware, software, libraries and other supporting software and business processes.

For example banks, health care institutions and insurance companies.

The elements of a legacy system



Legacy system layers



Legacy system replacement

- Legacy system replacement is **risky and expensive** so businesses continue to use these systems
- System replacement is risky for a number of reasons
  - Lack of complete system specification
  - Tight integration of system and business processes
  - Undocumented business rules embedded in the legacy system
  - New software development may be late and/or over budget

Tampereen yliopisto
Tampere University

code-for-a-living APRIL 20, 2020

Brush up your COBOL: Why is a 60 year old language suddenly in demand?

The suddenly strained unemployment systems often run on a 60-year-old programming language, COBOL. So, how can you learn it, make big bucks, and save lots of state agencies that need new code to deal with all the new government stimulus programs?

Charles R. Martin

[https://stackoverflow.blog/2020/04/20/brush-up-your-cobol-why-is-a-60-year-old-language-suddenly-in-demand/]

Once upon a time, when the world and computers were new, I was in an Associate's Degree program for Data Processing—there were no "computer science" programs then—in which I had to study accounting, math, statistics, and three computer languages: IBM/360 Basic Assembly Language, FORTRAN, and COBOL. By the 80's, students were being told that COBOL was a dead language, and no one was studying it any more.

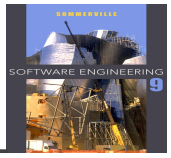
Now, in 2020, governments and banks are pleading for COBOL programmers, the language that wouldn't die.

The coronavirus pandemic is pushing health care systems to their limits, but it's also exposing the reliance of some states on very old technology. A side effect of that is COBOL programmers are back in demand.
The problem is, these systems are 40-year-old mainframes and rely on COBOL to keep them running.
"Not only do we need healthcare workers but given the legacy systems, we should add a page for COBOL computer skills because that's what we're dealing with".

US States Need COBOL Programmers to Help Them Handle Aging Systems

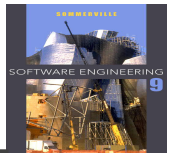
PCMag Follow
Apr 10 · 2 min read ★

Real-time systems



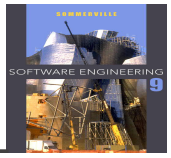
Real-time operating systems

Real-time operating systems



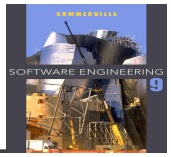
- Real-time operating systems are specialised operating systems which manage the processes in the RTS.
- Responsible for process management and resource (processor and memory) allocation.
- May be based on a standard kernel which is used unchanged or modified for a particular application.
- Do not normally include facilities such as file management.

Operating system components



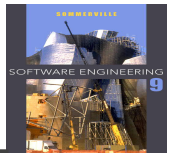
- Real-time clock
  - Provides information for process scheduling.
- Interrupt handler
  - Manages aperiodic requests for service.
- Scheduler
  - Chooses the next process to be run.
- Resource manager
  - Allocates memory and processor resources.
- Dispatcher
  - Starts process execution.

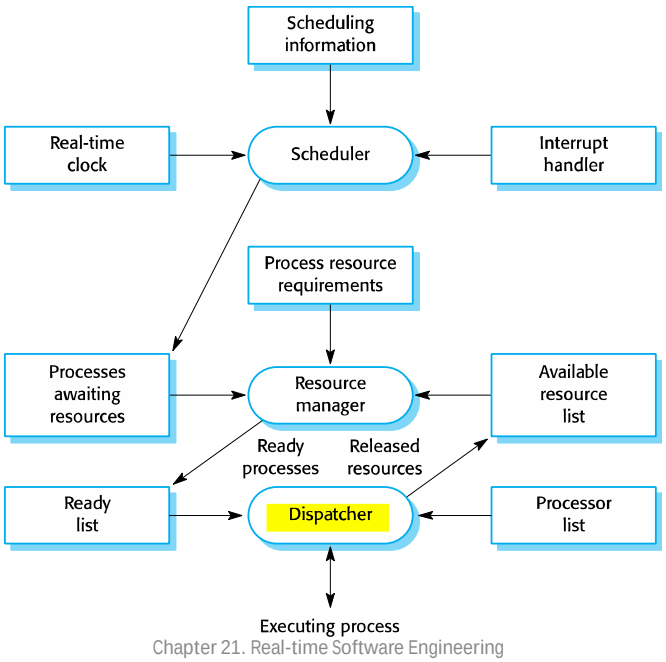
Non-stop system components



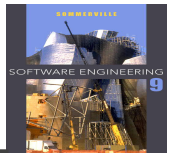
- Configuration manager
  - Responsible for the dynamic reconfiguration of the system software and hardware. Hardware modules may be replaced and software upgraded without stopping the systems.
- Fault manager
  - Responsible for detecting software and hardware faults and taking appropriate actions (e.g. switching to backup disks) to ensure that the system continues in operation.

Components of a real-time operating system



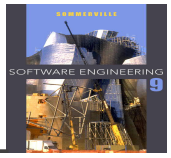


Process management



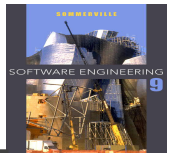
- Concerned with managing the set of concurrent processes.
- Periodic processes are executed at pre-specified time intervals.
- The RTOS uses the real-time clock to determine when to execute a process taking into account:
  - Process period - time between executions.
  - Process deadline - the time by which processing must be complete.

Process management



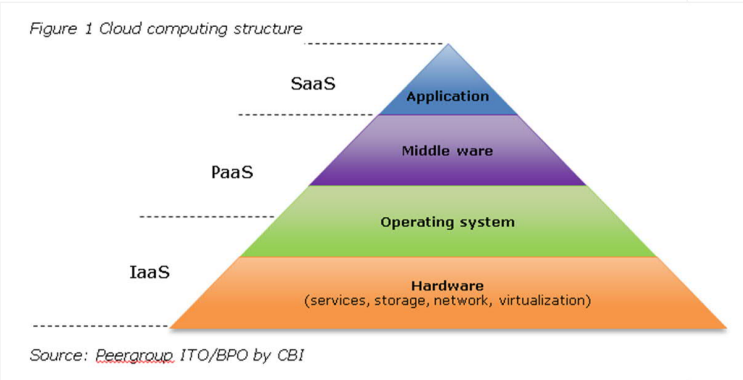
- The processing of some types of stimuli must sometimes take priority.
- Interrupt level priority. Highest priority which is allocated to processes requiring a very fast response.
- Clock level priority. Allocated to periodic processes.
- Within these, further levels of priority may be assigned.

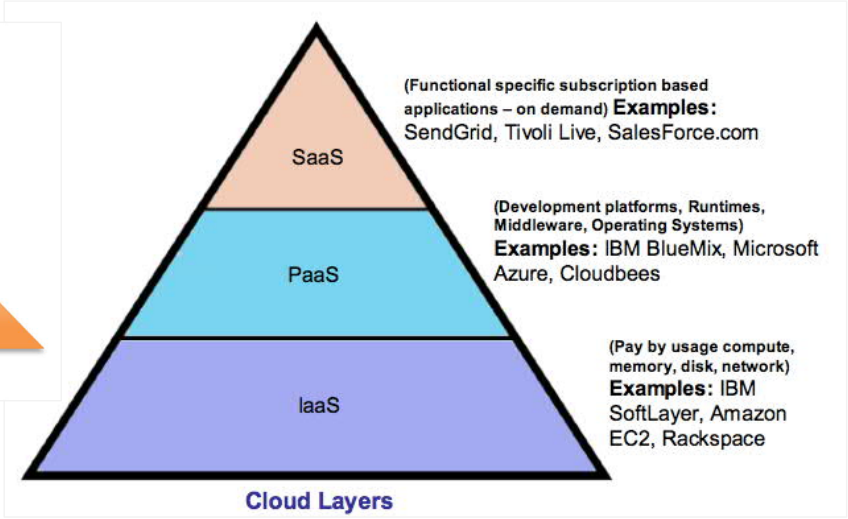
Interrupt servicing



- Control is transferred automatically to a pre-determined memory location.
- This location contains an instruction to jump to an interrupt service routine.
- Further interrupts are disabled, the interrupt serviced and control returned to the interrupted process.
- Interrupt service routines MUST be short, simple and fast. ISR

SaaS, PaaS, IaaS,... Cloud computing





SaaS = sw as a service
PaaS = platform as a service
IaaS = infrastructure as a service.

[www.ibm.com/blogs/cloud-computing/]

Tampereen yliopisto
Tampere University

SaaS vs PaaS vs IaaS: What's The Difference and How To Choose

software

platform

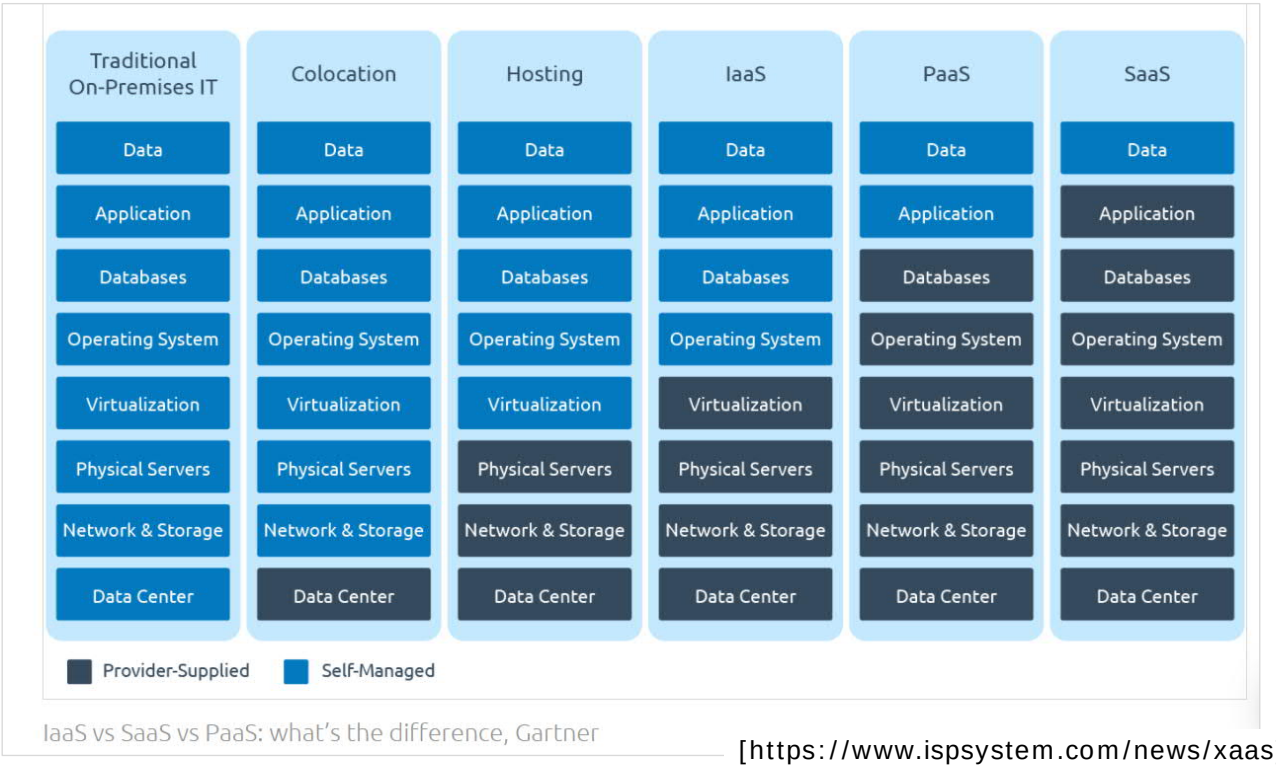
infrastructure

Common Examples of SaaS, PaaS, & IaaS

Platform Type	Common Examples
SaaS	Google Apps, Dropbox, Salesforce, Cisco WebEx, Concur, GoToMeeting
PaaS	AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift
IaaS	DigitalOcean, Linode, Rackspace, Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)

[www.bmc.com]

Tampereen yliopisto
Tampere University



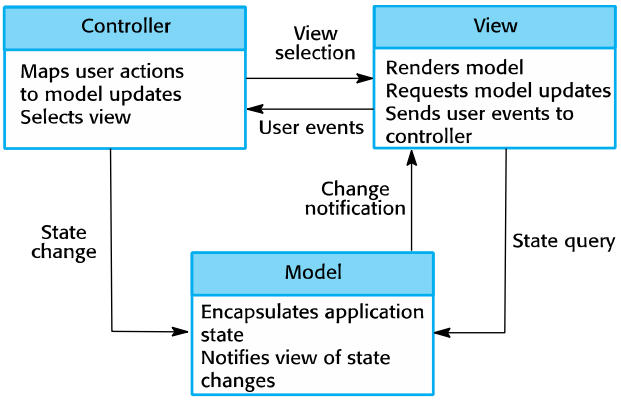
IaaS vs SaaS vs PaaS: what's the difference, Gartner

[https://www.ispsystem.com/news/xaas]

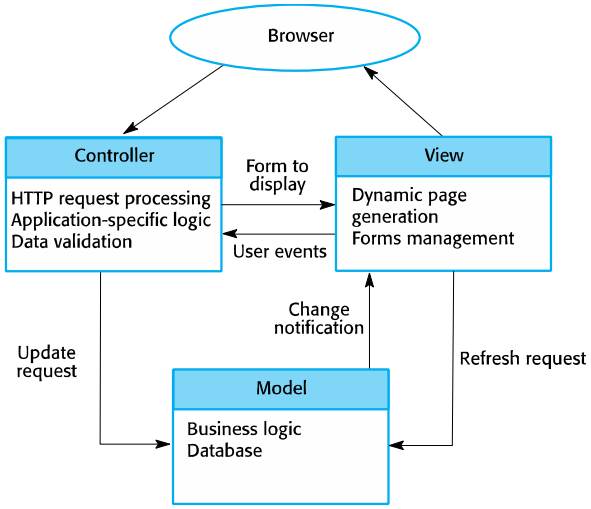
The Model-View-Controller (MVC) pattern

Name	MVC (Model-View-Controller)
Description	Separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other. The Model component manages the system data and associated operations on that data. The View component defines and manages how the data is presented to the user. The Controller component manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model. See Figure 6.3.
Example	Figure 6.4 shows the architecture of a web-based application system organized using the MVC pattern.
When used	Used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.
Advantages	Allows the data to change independently of its representation and vice versa. Supports presentation of the same data in different ways with changes made in one representation shown in all of them.
Disadvantages	Can involve additional code and code complexity when the data model and interactions are simple.

The organization of the Model-View-Controller



Web application architecture using the MVC pattern



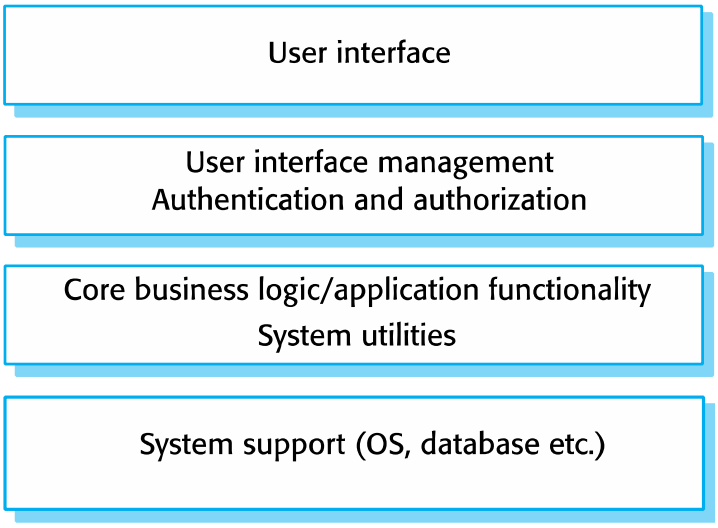
Layered architecture

- Used to model the interfacing of sub-systems.
- Organises the system into a set of layers (or abstract machines) each of which provide a set of services.
- **Supports the incremental development** of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.
- However, often artificial to structure systems in this way.

The Layered architecture pattern

Name	Layered architecture
Description	Organizes the system into layers with related functionality associated with each layer. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system. See Figure 6.6.
Example	A layered model of a system for sharing copyright documents held in different libraries, as shown in Figure 6.7.
When used	Used when building new facilities on top of existing systems; when the development is spread across several teams with each team responsibility for a layer of functionality; when there is a requirement for multi-level security.
Advantages	Allows replacement of entire layers so long as the interface is maintained. Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system.
Disadvantages	In practice, providing a clean separation between layers is often difficult and a high-level layer may have to interact directly with lower-level layers rather than through the layer immediately below it. Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer.

A generic layered architecture



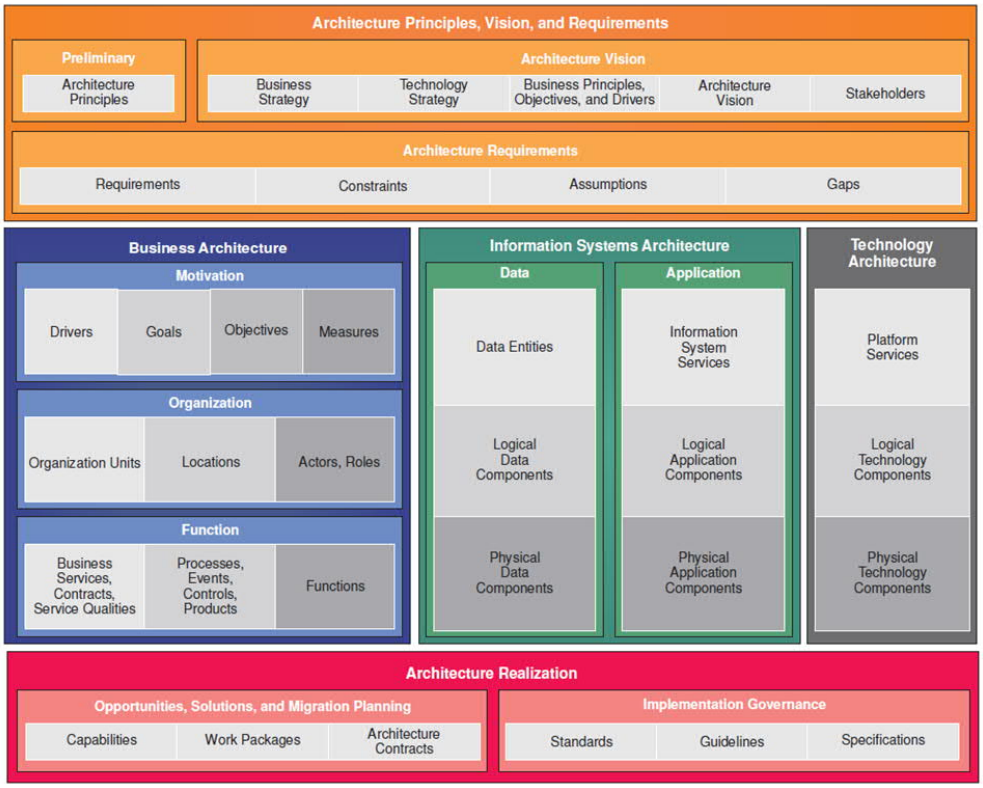
Tampereen yliopisto
Tampere University

Architecture Framework

Tampereen yliopisto
Tampere University

TOGAF =
The Open Group
Architecture Framework

(FI: kokonaisarkkitehtuuri-viitekehys)



[https://guozspace.com/2013/04/28/togaf-9-1-content-metamodel-overview/]

Tampereen yliopisto
Tampere University

Architect roles

Roles

Difference between IT Architect Roles

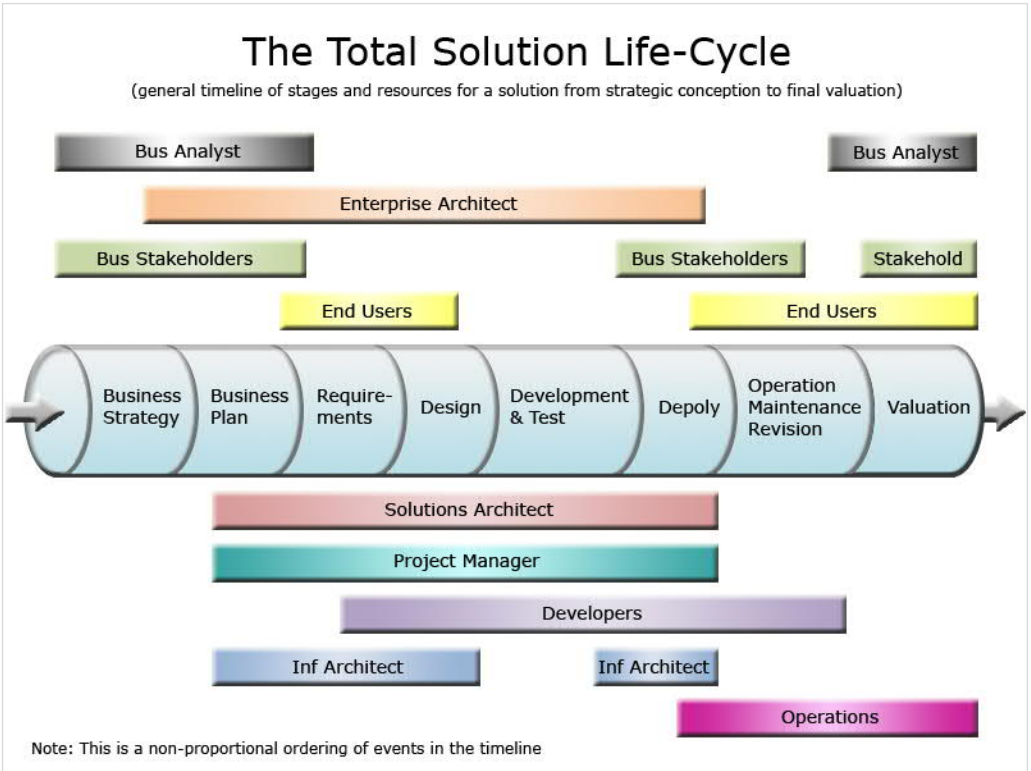
Enterprise Architect	Solutions Architect	Technical Architect
▪ Enterprise Architect (EA) is a planning role that is responsible for identifying the future state of an organization's IT environment and engage wherever and whomever necessary to help guide project teams to deliver toward it. ▪ An EA would ensure IT investments are aligned with business strategy. ▪ Responsible for strategic thinking, roadmaps, principles and governance of the entire enterprise. ▪ Enterprise Architect defines which problem need a solution	▪ Solutions Architect (SA) focuses on delivery of a particular solution. ▪ The SA is responsible for implementing a strategic IT program within the framework laid down by the enterprise architecture (EA) team. ▪ Solutions Architect (SA) is assigned to ensure technical integrity and consistency of the solution on every stage of its life-cycle ▪ Solutions Architect translates a problem to a solution	▪ Technical Architect is usually a technology specialist in a particular technology or group of interrelated technologies. ▪ Job titles vary for this role and they may also include Infrastructure Architect, Domain Architect, Application Architect (Java Architect, .Net Architect), Network Architect, Security Architect . ▪ Technical Architect works within a solution

[https://image.slidesharecdn.com/ss-180127053749/95/solution-architecture-8-638.jpg?cb=1517031523]

8 http://blog.danovich.com.au/2012/03/24/difference-between-it-architect-roles/ | http://sysarchitect.com/2011/09/19/differences-between-architecture-roles/ | http://sysarchitect.com/2011/09/19/differences-between-architecture-roles/

Tampereen yliopisto
Tampere University

The total solution life cycle includes the business strategy and business planning activities that precede the software development life cycle (SDLC), as well as the deployment and ongoing operations that follow. We examine the tools used to create business strategies, develop and implement solutions, and evaluate solution effectiveness. A solutions architect that understands the total solution life cycle can make better decisions than a peer who understands only the SDLC.

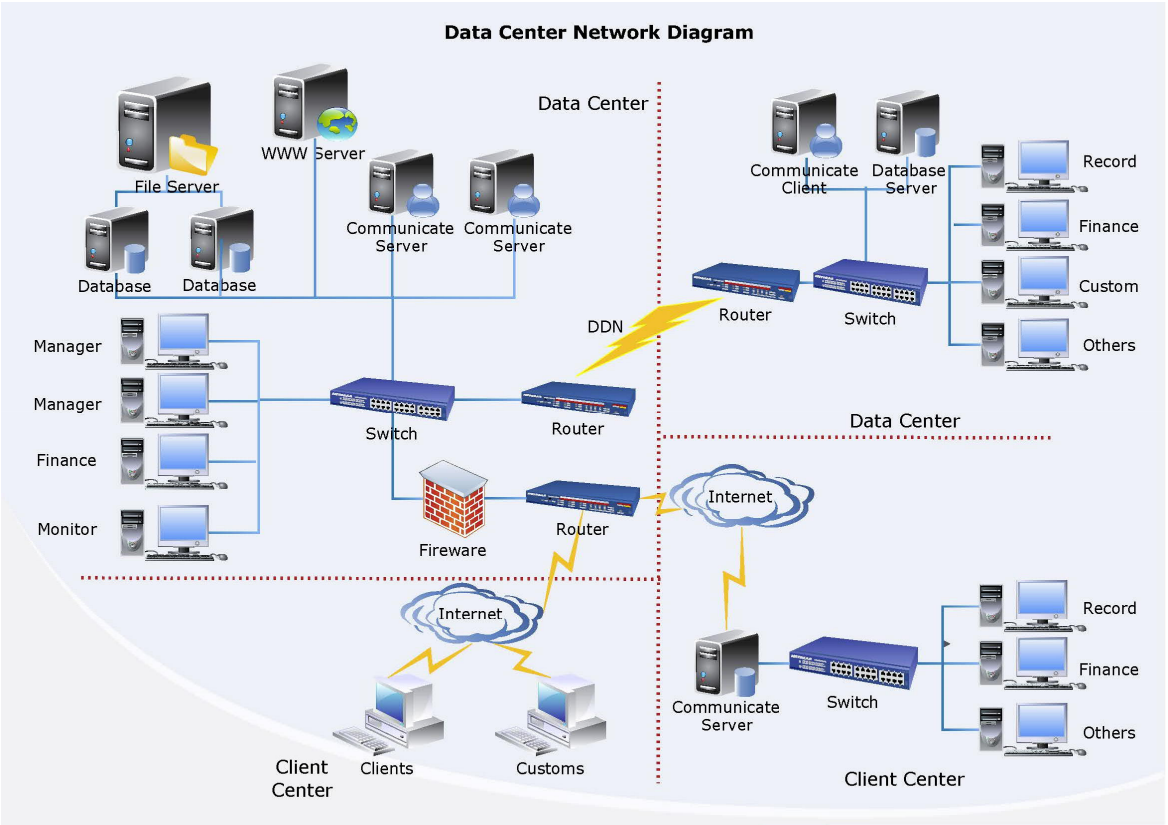


[https://docs.microsoft.com/en-us/previous-versions/bb756611(v=msdn.10)?redirectedfrom=MSDN]

Architecture matters [Sommerville, 2016]

A software architecture is a description of how a software system is organized. Properties of a system such as performance, security, and availability are influenced by the architecture used.

- Architectures may be documented from several different perspectives or views. Possible views include a **conceptual** view, a **logical** view, a **process** view, a **development** view, and a **physical** view.
- Architectural patterns are a means of reusing knowledge about generic system architectures. They describe the architecture, explain when it may be used, and point out its advantages and disadvantages.
- Commonly used Architectural patterns include model-view-controller, layered architecture, repository, client–server, and pipe and filter.
- Generic models of application systems architectures help us understand the operation of applications, compare applications of the same type, validate application system designs, and assess large-scale components for reuse.
- Language processing systems are used to translate texts from one language into another and to carry out the instructions specified in the input language. They include a translator and an abstract machine that executes the generated language.



Logical view

2.2 Software Architecture Description

Stakeholders have different viewpoints and views

(replay picture from previous lectures)

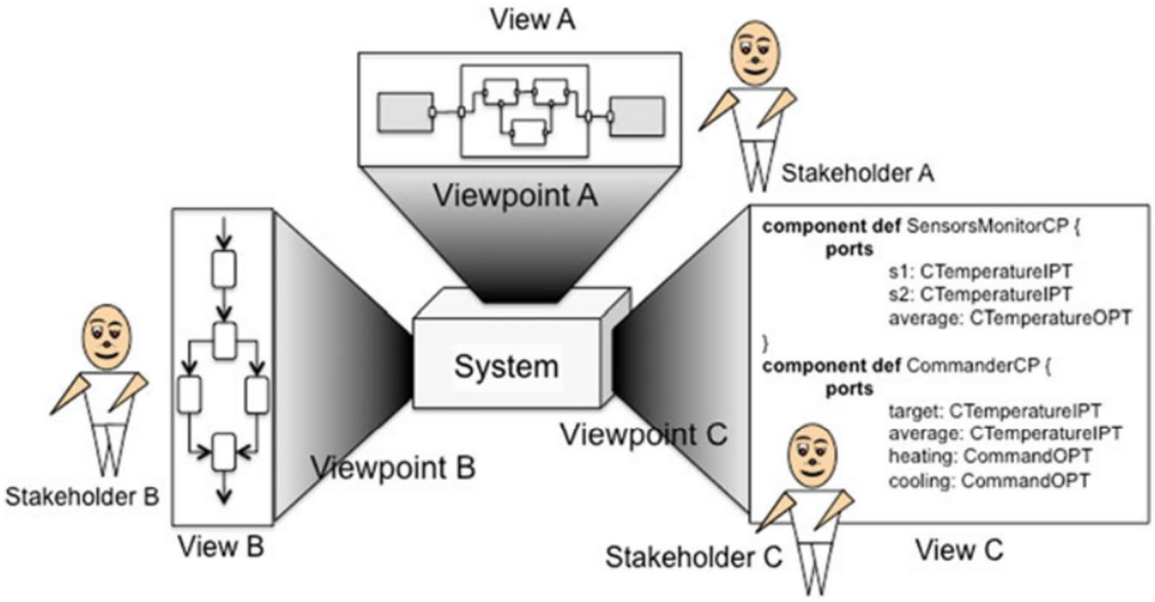
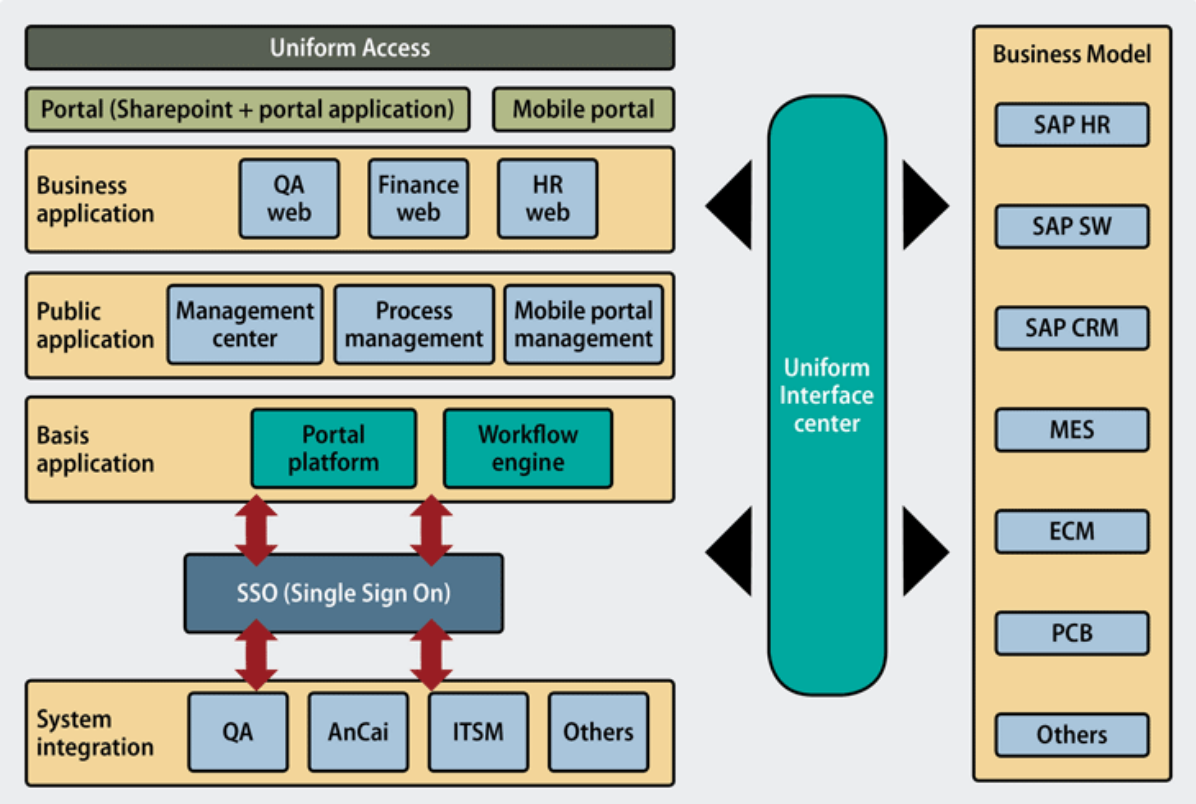


Fig. 2.5 Viewpoints and views

[Software Architecture in Action, 2016]

Tampereen yliopisto
Tampere University

In a typical example of enterprise architecture, all elements are included in the overall schema, from business models to business applications to system integration workflow.

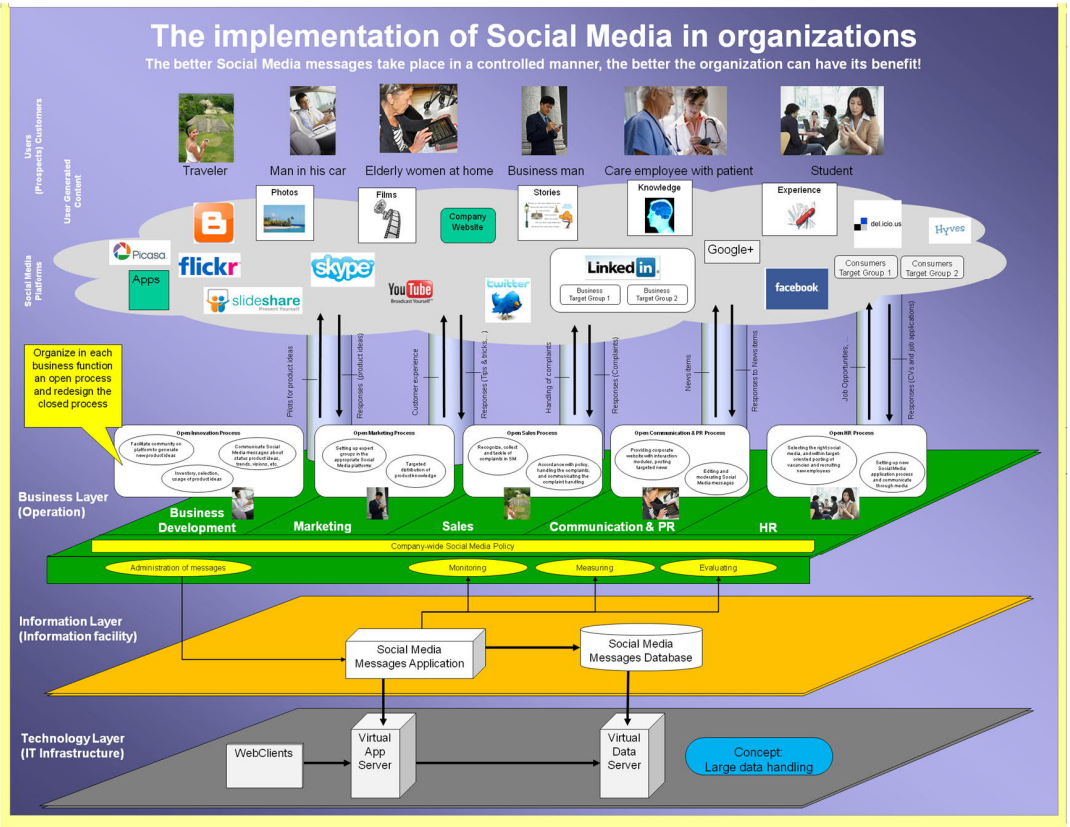


[www.datamation.com]

Tampereen yliopisto
Tampere University

Architectural view;

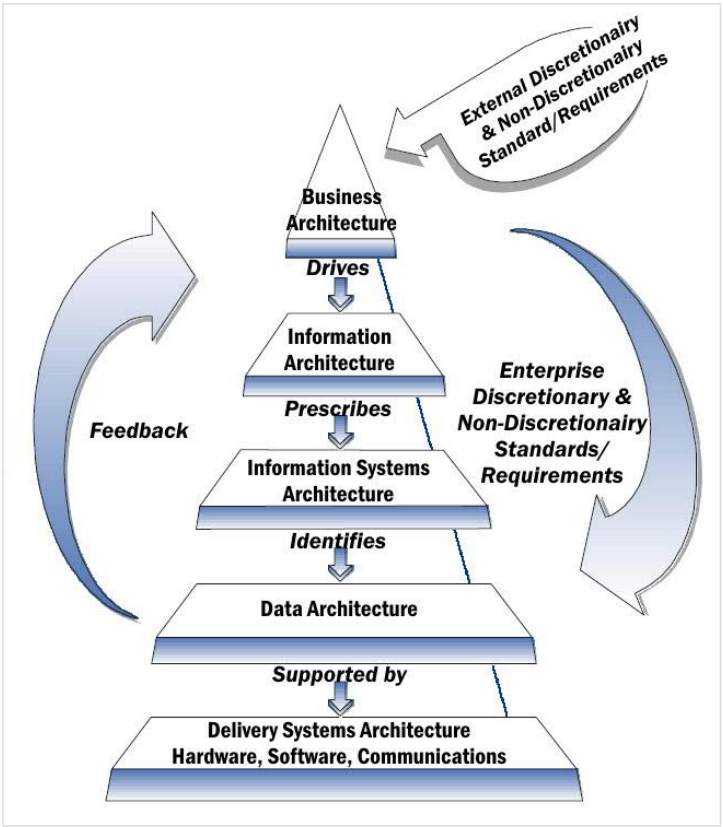
”Social Media 2.0;
The Way It Works”.



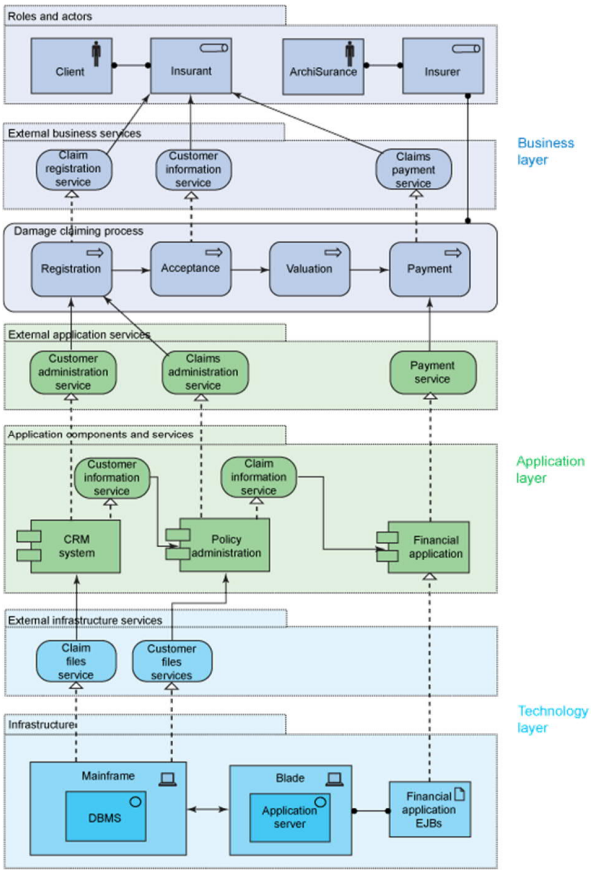
[www.dragon1.com/blogs/markpaauwe/can-visual-enterprise-architecture-based-on-dragon1-be-done-by-anyone]

Tampereen yliopisto
Tampere University

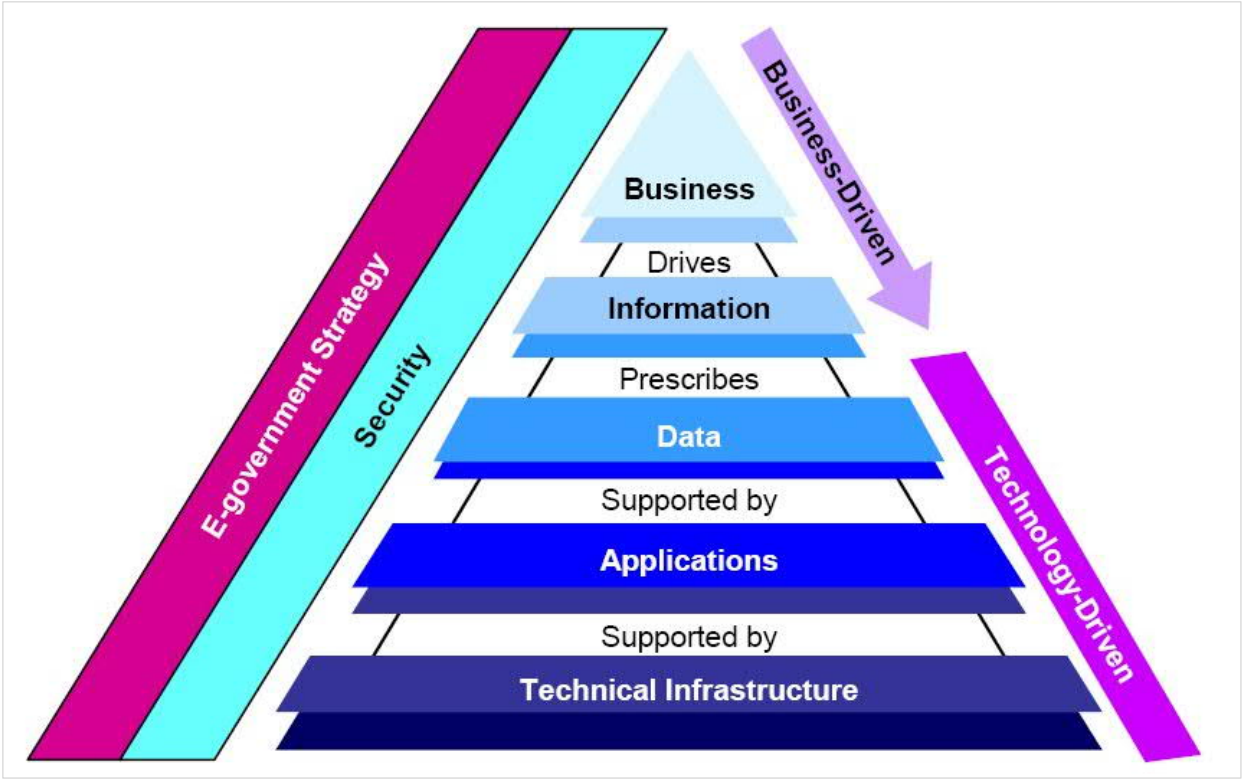
NIST (National Institute of Standards and Technology) Enterprise Architecture

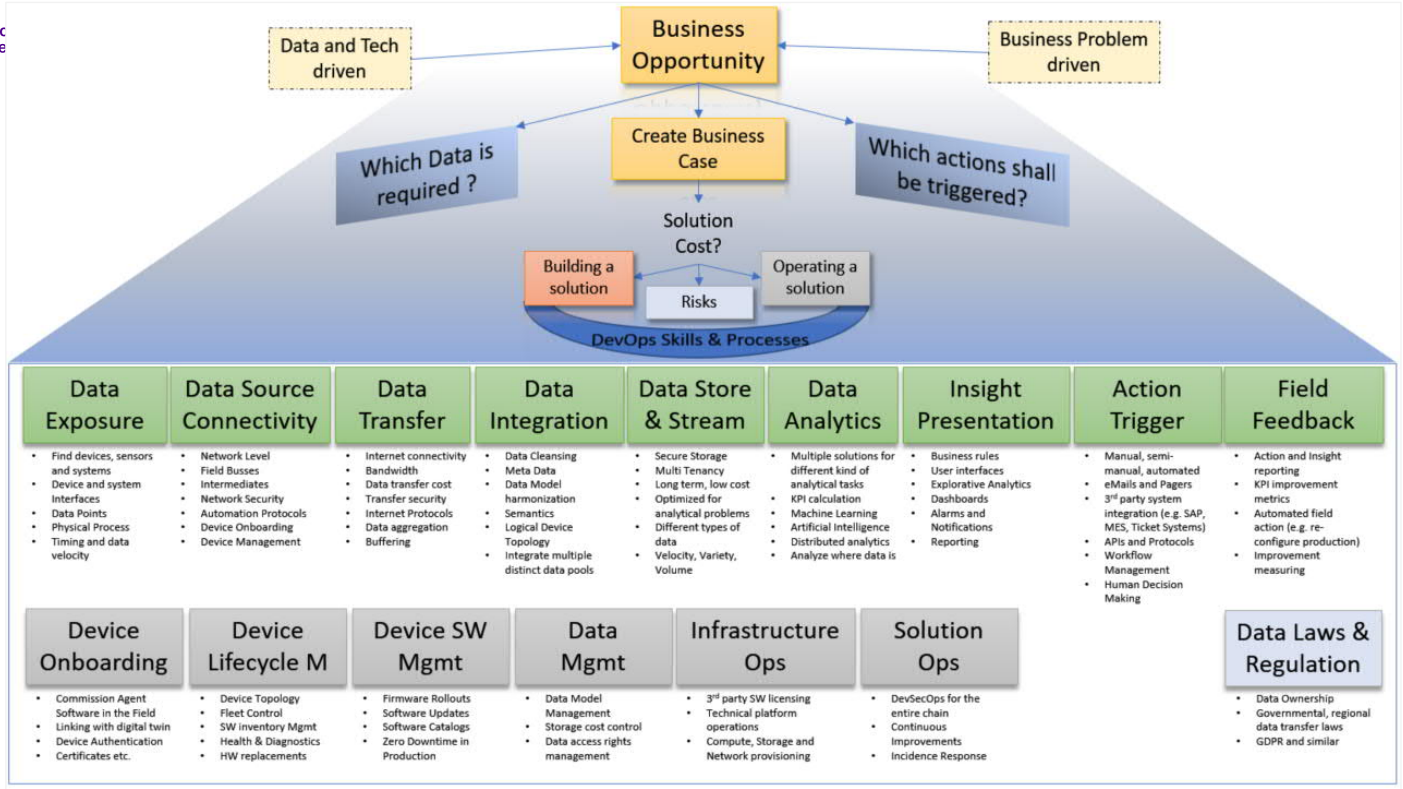


IBM architecture



FDIC (Federal Deposit Insurance Company) enterprise architecture framework





[https://ideal-digital.org/2019/04/17/a-taxonomy-of-industrial-iot-solution-building/]

[softwarearchitectureinpractice/]

Definition: The software architecture of a program or computing system is the structure or structures of the system, which comprise

- software elements,
- the externally visible properties of those elements,
- and the relationships among them.

- "Externally visible" properties are those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.

Implications of this definition:
Architecture defines software elements

- how the elements relate to each other
- an architecture is foremost an abstraction of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements. (only public part)

Systems comprise more than one structure

- no one structure can claim to be the architecture
- Relationships and elements might be
  - runtime related ("send data", processes)
  - nonruntime related ("submodel of", "inherits from", class)

Every computing system with software has a software architecture

- it does not necessarily follow that the architecture is known to anyone.
- An architecture can exist independently of its description or specification

The behavior of each element is part of the architecture

- allows elements to interact with each other
- Add specification and properties to the elements and relationships

The definition is indifferent as to whether the architecture for a system is a good one or a bad one

- Eveluating the SW architecture in the context of usa.

Zachman's 6x6 matrix.

The Zachman Framework is an enterprise ontology and is a fundamental structure for Enterprise Architecture which provides a formal and structured way of viewing and defining an enterprise.

Generic Classification Structure of Design Artifacts

	What	How	Where	Who	When	Why	
Planner							Scope
Owner							Concepts
Designer							Logic
Builder							Physics
Imple-menter							Technology
Operator		THE ENTERPRISE					Product
	Material	Process	Geometry	Instructions.	Timing	Objectives	

[https://www.zachman.com/images/ZI_PIcs/Generic_Framework.png]

The Scope of Focus in Zachman EA Framework

Views	What	How	Where	Who	When	Why
	Entity/Data	**Function**	**Network**	**People**	**Time**	**Motivation**
Business Environment (Planner)	**Info/Data Architecture** (scope of focus)	**Enterprise Architecture** (scope of focus)				
Capabilities, Functions, Services (Owner)						
Logical Systems (Designer)		**Solution Architectures** (scope of focus)				
Physical Systems (Builder)						
Technology References		**Technology Architectures** (scope of focus)				
Technology Details Specification						

Yan Zhao, Ph.D, ArchiTech Consulting LLC, 2011

Name	Advantages	Disadvantages
Layered	A lower layer can be used by different higher layers. Layers make standardization easier as we can clearly define levels. Changes can be made within the layer without affecting other layers.	Not universally applicable. Certain layers may have to be skipped in certain situations.
Client-server	Good to model a set of services where clients can request them.	Requests are typically handled in separate threads on the server. Inter-process communication causes overhead as different clients have different representations.
Master-slave	Accuracy - The execution of a service is delegated to different slaves, with different implementations.	The slaves are isolated: there is no shared state. The latency in the master-slave communication can be an issue, for instance in real-time systems. This pattern can only be applied to a problem that can be decomposed.
Pipe-filter	Exhibits concurrent processing. When input and output consist of streams, and filters start computing when they receive data. Easy to add filters. The system can be extended easily. Filters are reusable. Can build different pipelines by recombining a given set of filters	Efficiency is limited by the slowest filter process. Data-transformation overhead when moving from one filter to another.
Broker	Allows dynamic change, addition, deletion and relocation of objects, and it makes distribution transparent to the developer.	Requires standardization of service descriptions.
Peer-to-peer	Supports decentralized computing. Highly robust in the failure of any given node. Highly scalable in terms of resources and computing power.	There is no guarantee about quality of service, as nodes cooperate voluntarily. Security is difficult to be guaranteed. Performance depends on the number of nodes.
Event-bus	New publishers, subscribers and connections can be added easily. Effective for highly distributed applications.	Scalability may be a problem, as all messages travel through the same event bus
Model-view-controller	Makes it easy to have multiple views of the same model, which can be connected and disconnected at run-time.	Increases complexity. May lead to many unnecessary updates for user actions.
Blackboard	Easy to add new applications. Extending the structure of the data space is easy.	Modifying the structure of the data space is hard, as all applications are affected. May need synchronization and access control.
Interpreter	Highly dynamic behavior is possible. Good for end user programmability. Enhances flexibility, because replacing an interpreted program is easy.	Because an interpreted language is generally slower than a compiled one, performance may be an issue.

[https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013]

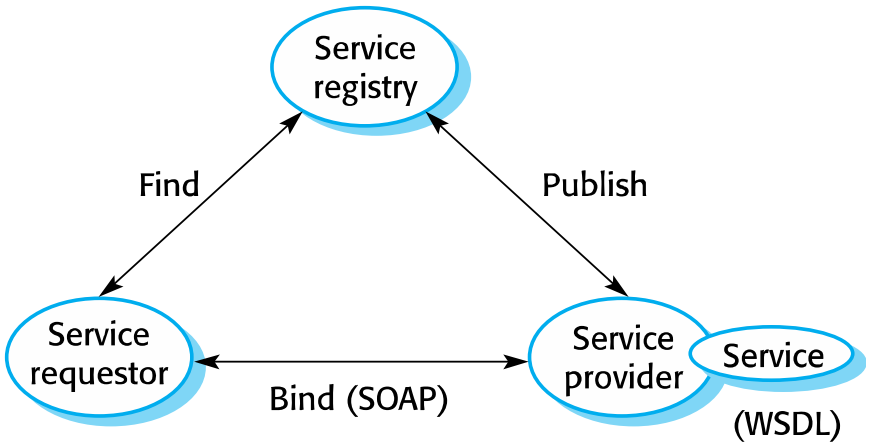
Tampereen yliopisto
Tampere University

Service-oriented architecture

Service-oriented architectures

- A means of **developing distributed systems** where the components are stand-alone services
- Services may execute on different computers from different service providers
- Standard protocols have been developed to support service communication and information exchange

Service-oriented architecture



Service - oriented application

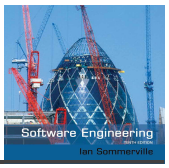
Benefits of SOA

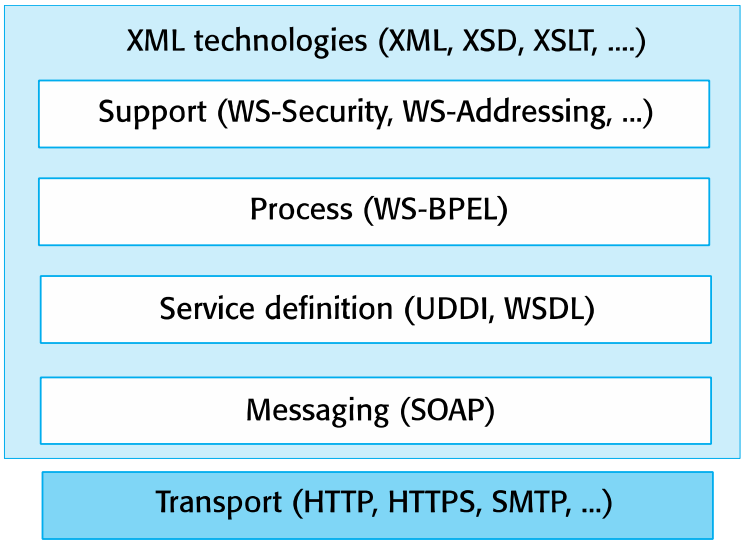
- Services can be provided locally or outsourced to external providers
- Services are language-independent
- Investment in legacy systems can be preserved
- Inter-organisational computing is facilitated through simplified information exchange

Key standards

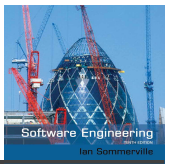
- SOAP
  - A message exchange standard that supports service communication
- WSDL (Web Service Definition Language)
  - This standard allows a service interface and its bindings to be defined
- WS-BPEL
  - A standard for workflow languages used to define service composition

Web service standards





Service-oriented software engineering



- Existing approaches to software engineering have to evolve to reflect the service-oriented approach to software development
  - Service engineering. The development of dependable, reusable services
    - Software development for reuse
  - Software development with services. The development of dependable software where services are the fundamental components
    - Software development with reuse.

RESTful web services

- Current web services standards have been criticized as ‘heavyweight’ standards that are over-general and inefficient.
- REST (REpresentational State Transfer) is an architectural style based on transferring representations of resources from a server to a client.
- This style underlies the web as a whole and is simpler than SOAP/WSDL for implementing web services.
- RESTful services involve a lower overhead than so-called ‘big web services’ and are used by many organizations implementing service-based systems.

Tampereen yliopisto
Tampere University

Now the additional L6
extra slides set ends here

Tampereen yliopisto
Tampere University

Now the additional L6
extra slides set ends here